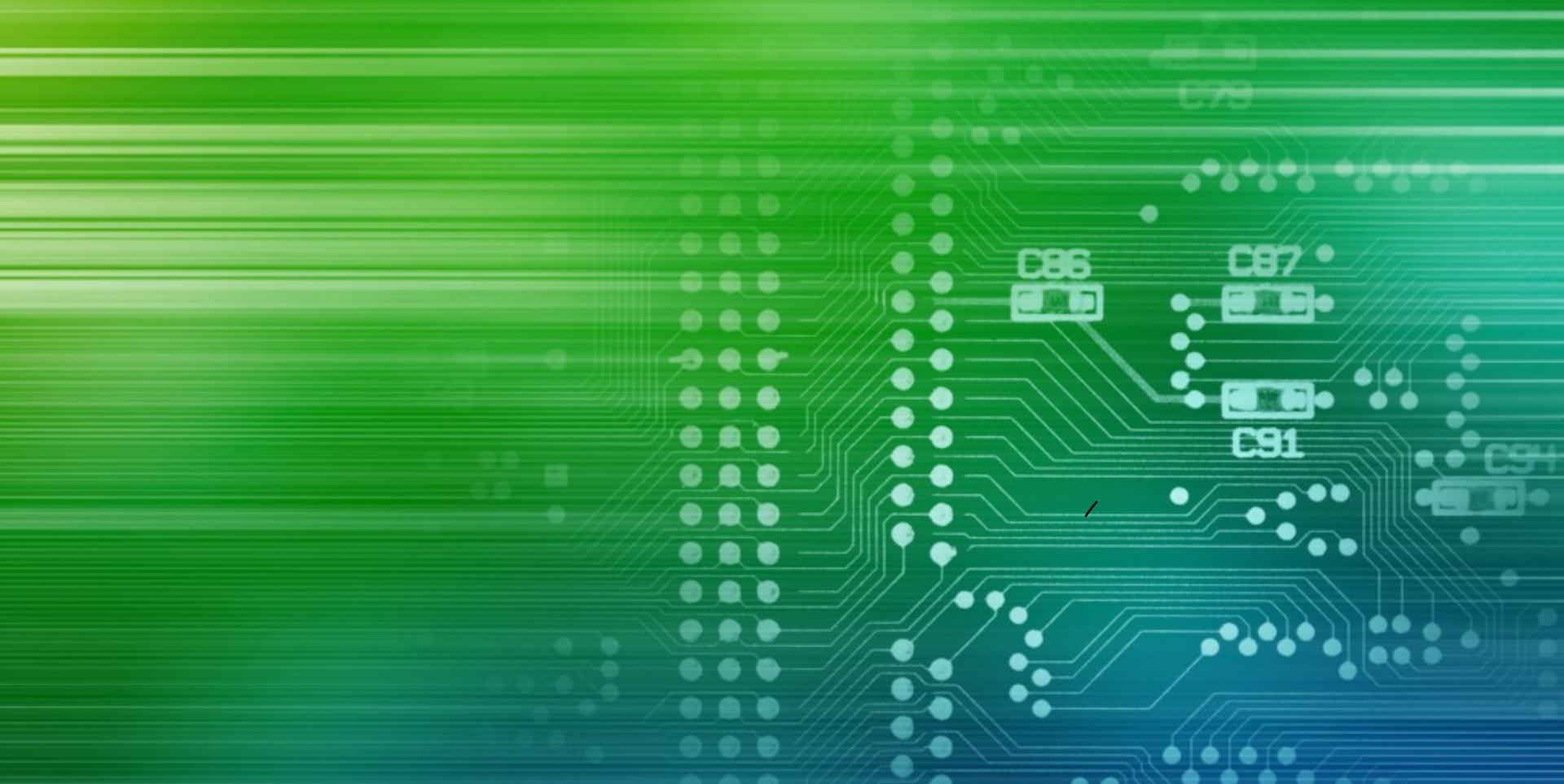


STRUCTURA ŞI ORGANIZAREA CALCULATOARELOR

Curs 1

Cuprins

- Obiective
- Cadre didactice
- Organizare
- Bibliografie recomandată
- Înţelegerea arhitecturilor interne
- CISC / RISC
- Activitate pentru laborator

Obiective

- Să-şi însuşească noţiunile de structură și organizare a sistemelor de calcul cu microprocesoare din familia 80X86 (CISC)
- Să realizeze proiectarea şi sinteza unui microcalculator cu procesor RISC sau CISC
- Arhitectura RISC - studiu de caz pe microcontrolerul AVR ATmega16

Cadre didactice

- Curs
  - **şl. dr. ing. Alexandru Bârleanu**
    - alexandru-theodor-cristian.barleanu@academic.tuiasi.ro
    - cabinet C4-1, tel. +40-232-278680 / int. 1349
- Laborator
  - **şl. dr. ing. Alexandru Bârleanu**
  - **drd. ing. Cristian Axinte**

Organizare

- Curs: 2 ore pe saptămână
- Laborator: 2 ore pe saptămână
- Evaluare finală: examen (practic și teoretic)
- Credite: 5

Organizare

- *Discipline anterioare*
  - **Arhitectura Sistemelor de Calcul**
  - Proiectare Logică
  - Electronică digitală
  - **Programarea Calculatoarelor**
  - Structuri de Date

- *Discipline prezente*
  - **Proiectarea cu Microprocesoare**
  - **Proiectarea Sistemelor Digitale**

Organizare

- Evaluare:
  - **Continuă:**
    - Curs (***NC***): referate despre subiectele studiate
    - Laborator (***NL***): **(minim 5)**
      - Teste: 2 teste anunţate din materia studiată;
      - Temă de casă: 1 problemă propusă spre rezolvare
      - Se acordă puncte pentru implicarea la laborator cu rezultate notabile
  - **Finală:**
    - examen (EX): test grilă din materia prezentată la curs şi laborator

NEX = RC*Prc - RG*Prc/3 + off

☺ ☹ ☺ v ☺

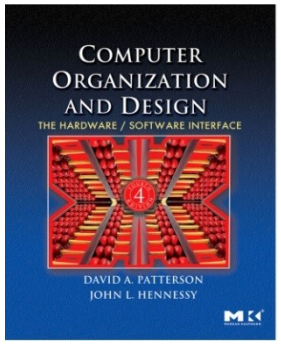
- ! (***NPR***) - se poate propune un set de teme de proiectare (începând din săptămâna 6) dintre care grupe de câte maxim 3 studenţi aleg şi rezolvă o problemă. Rezolvarea problemelor este susţinută cu o prezentare la sfârşitul semestrului.

NSOC = 0.1* NC + 0.3* NL + 0.4* NEX + 0.1*NPR

Organizare

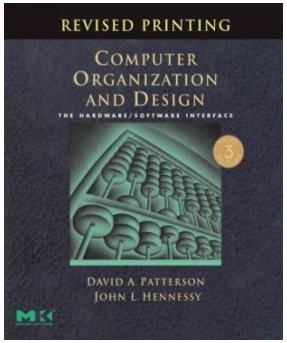
- Echipamente pentru activitatea de laborator
  - **Microcontrolerul AVR ATmega16 şi debuggerul JTAG ICE**
  - **Mediul de dezvoltare AVR Studio şi compilatorul IAR pentru AVR**

Bibliografie recomandată



- Computer Organization and Design, Four th Edition
  ; David A. Patterson; John L. Hennessy, Elsevier Science, 2008

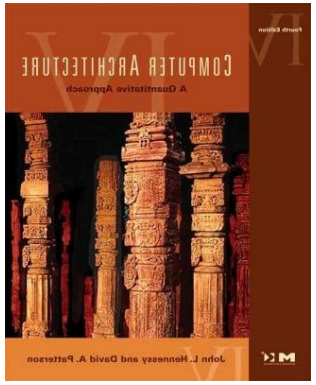
  ISBN-13: 9780123744937 **ISBN-10:** 0123744938



- Computer Organization and Design, David A. Patterson; John L. Hennessy, Elsevier Science, 2007

  ISBN-13: 9780123706065 **ISBN-10:** 0123706068

Bibliografie recomandată



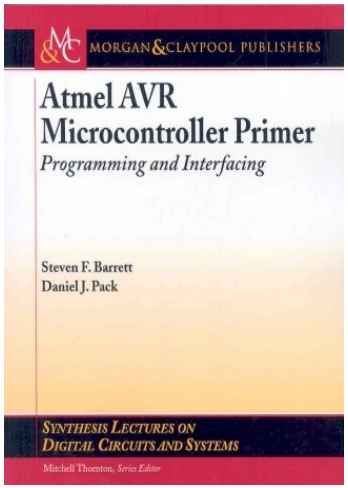
- Computer Architecture , John L. Hennessy; David A. Patterson , Elsevier Science, 2006

  ISBN-13: 9780123704900 **ISBN-10:** 0123704901

- Microcontroller Systems Engineering ; Bert van Dam , Elektor Electronics, 2009

  ISBN-13: 9780905705750 **ISBN-10:** 0905705750

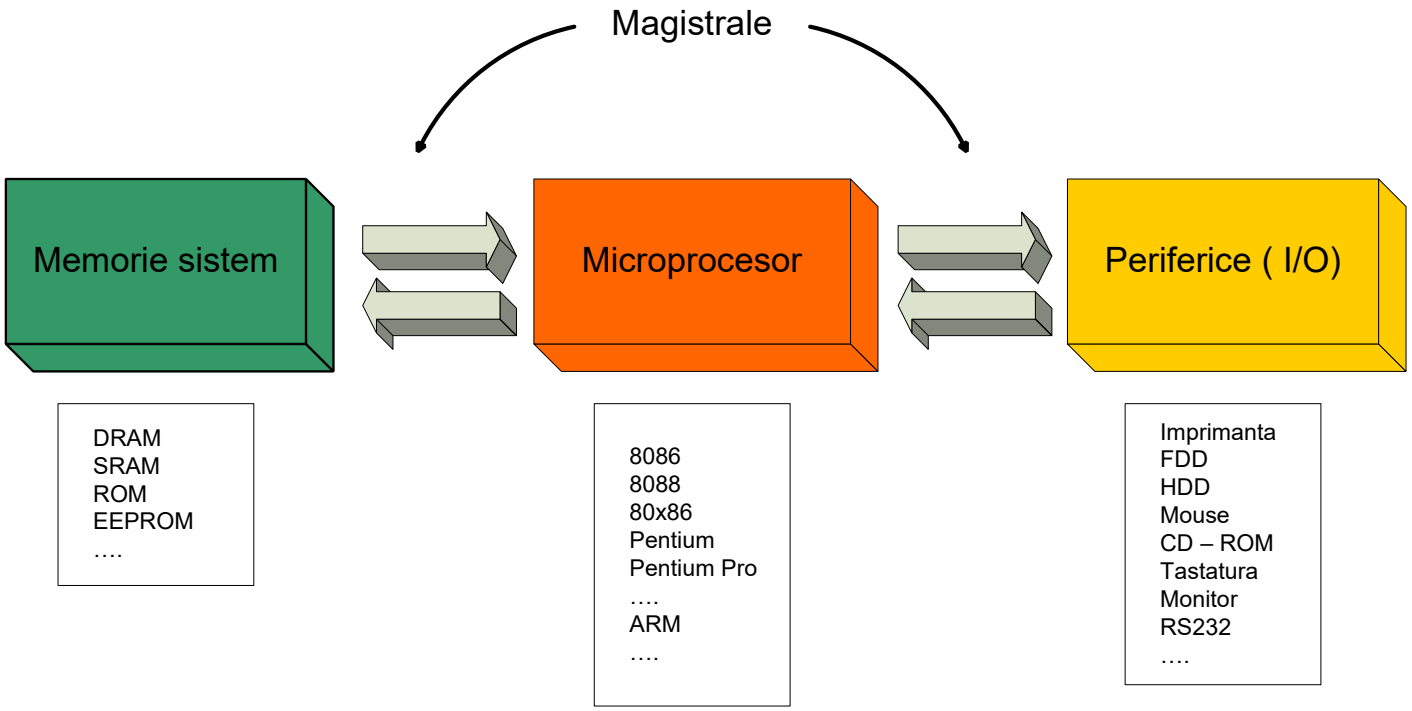
Bibliografie recomandată



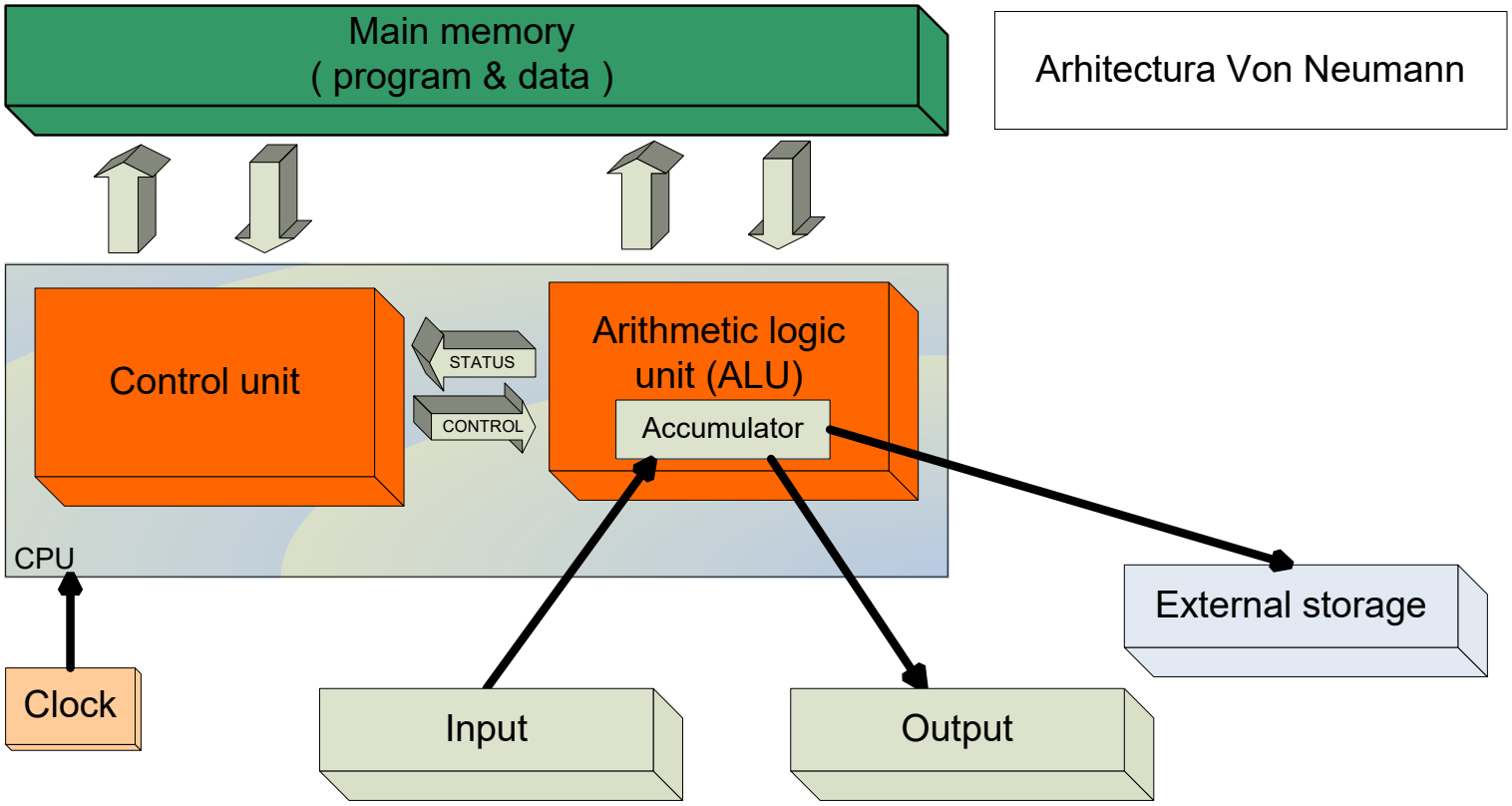
- Atmel Avr Microcontroller Primer Programming and Interfacing; M.D. Barrett, Steven; Daniel Pack; Mitchell (EDT) Thornton, Morgan & Claypool, 2007

ISBN-13: 9781598295412 **ISBN-10:** 1598295411

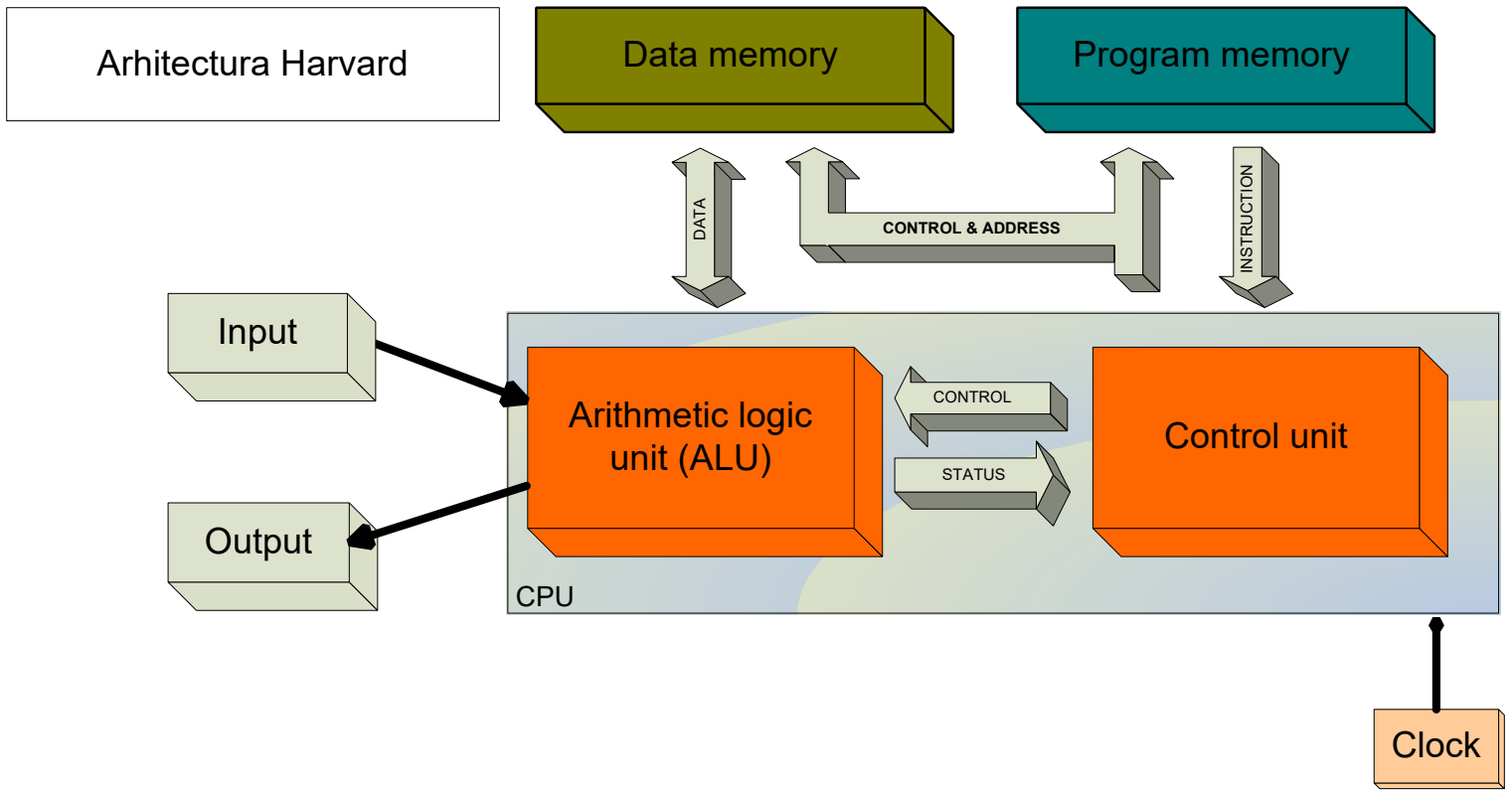
Structura şi organizarea calculatoarelor



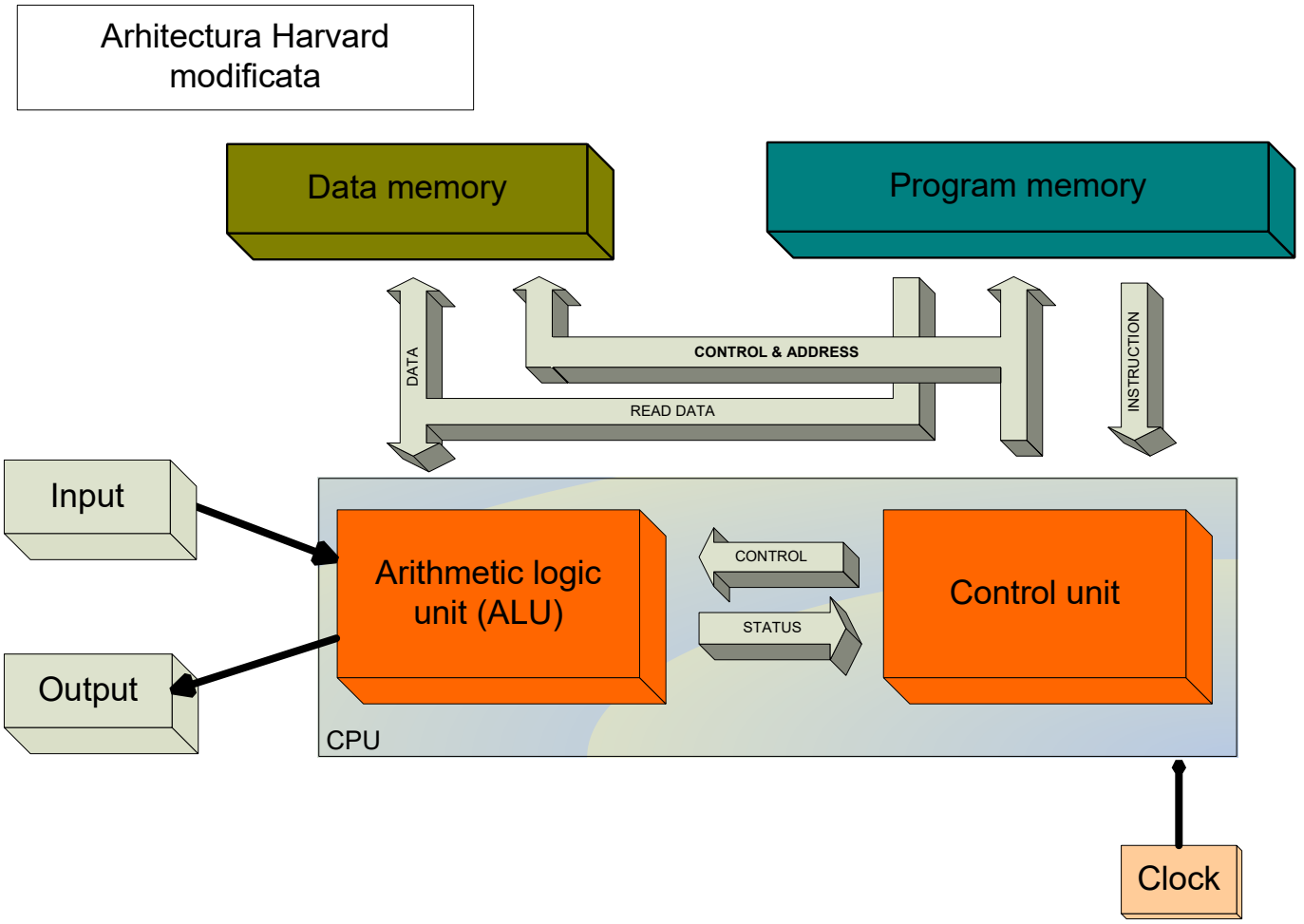
Structura şi organizarea calculatoarelor



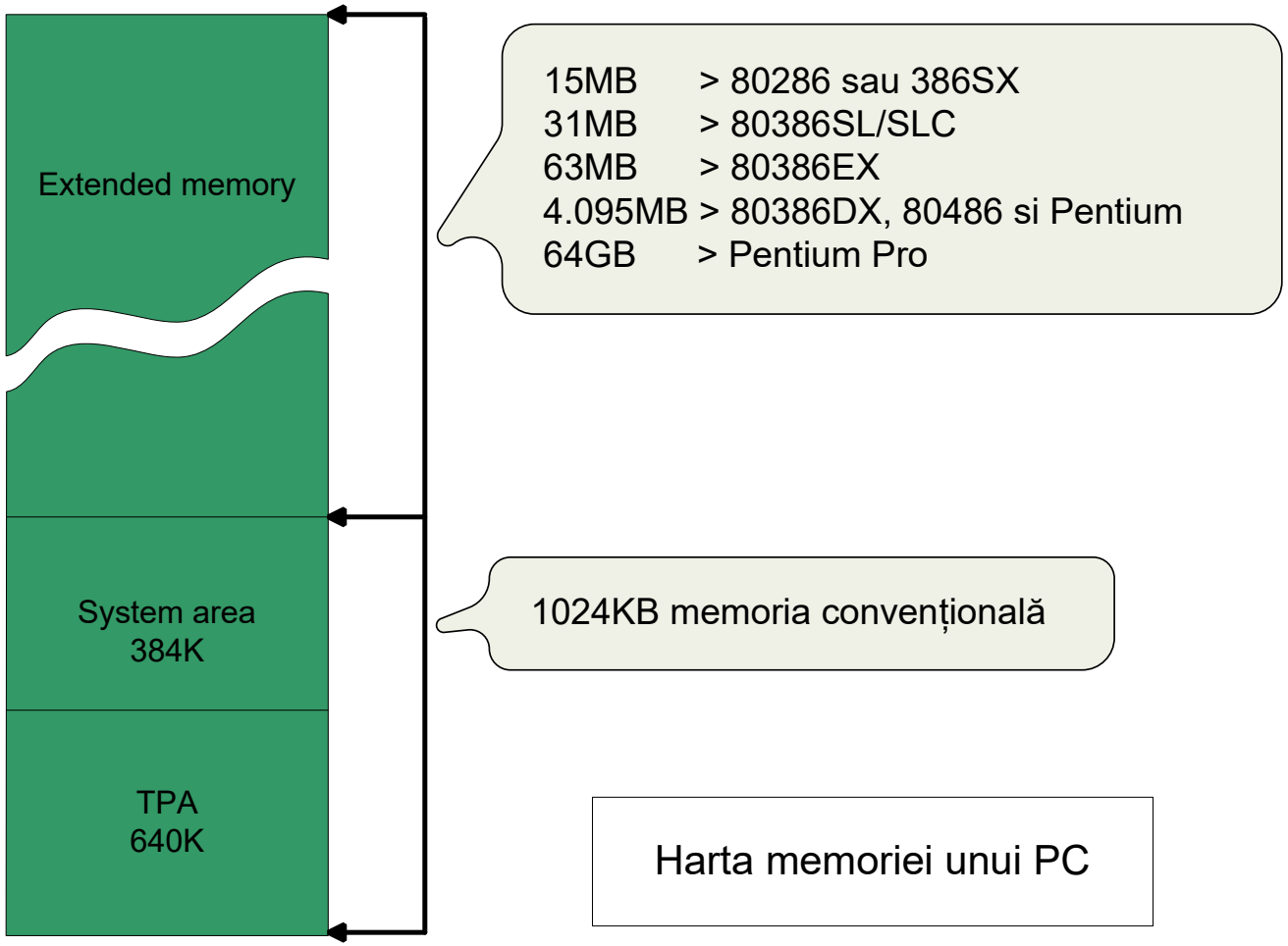
Structura şi organizarea calculatoarelor



Structura şi organizarea calculatoarelor



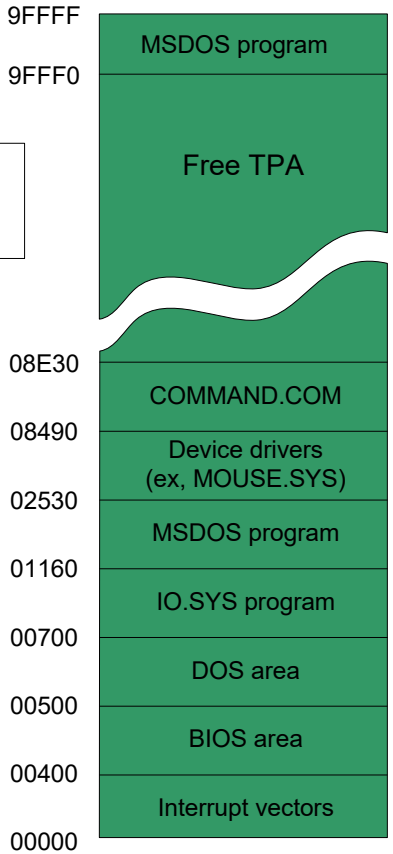
Structura şi organizarea calculatoarelor



Harta memoriei unui PC

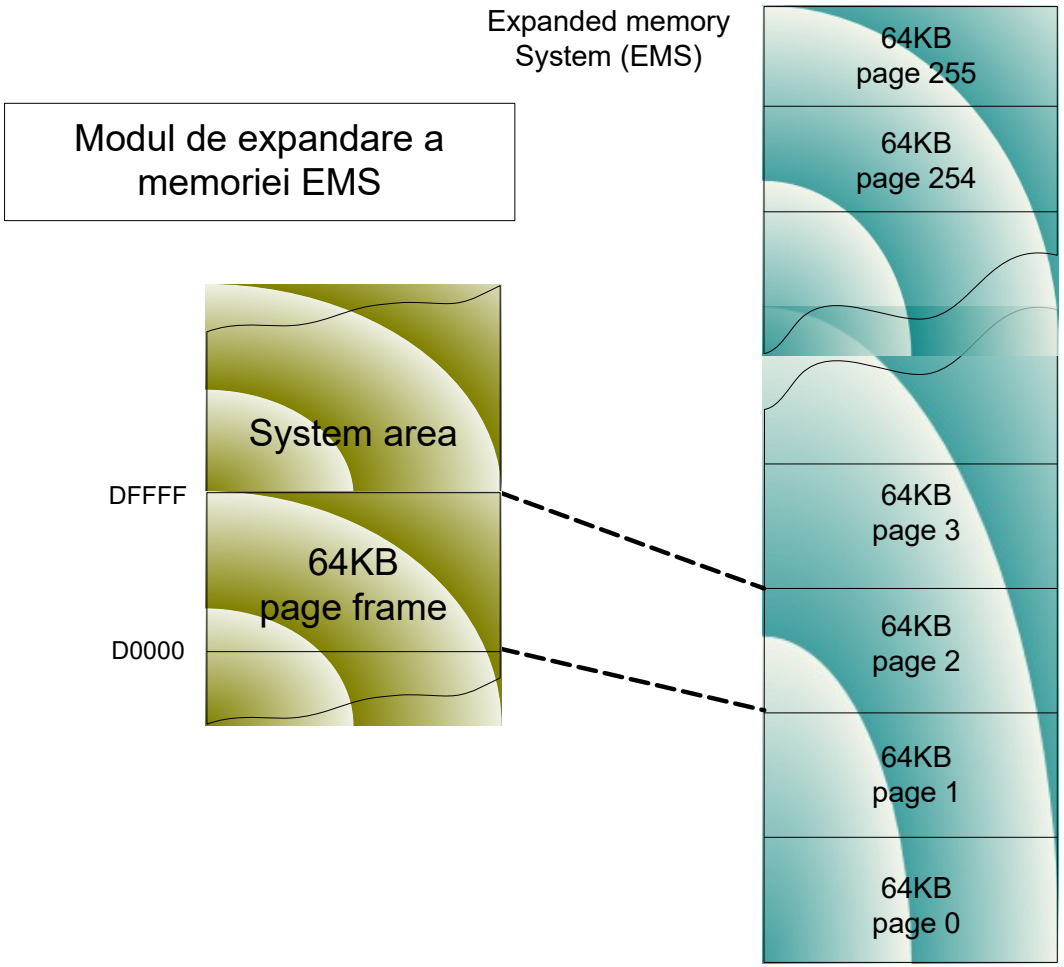
Structura şi organizarea calculatoarelor

Harta memoriei TPA



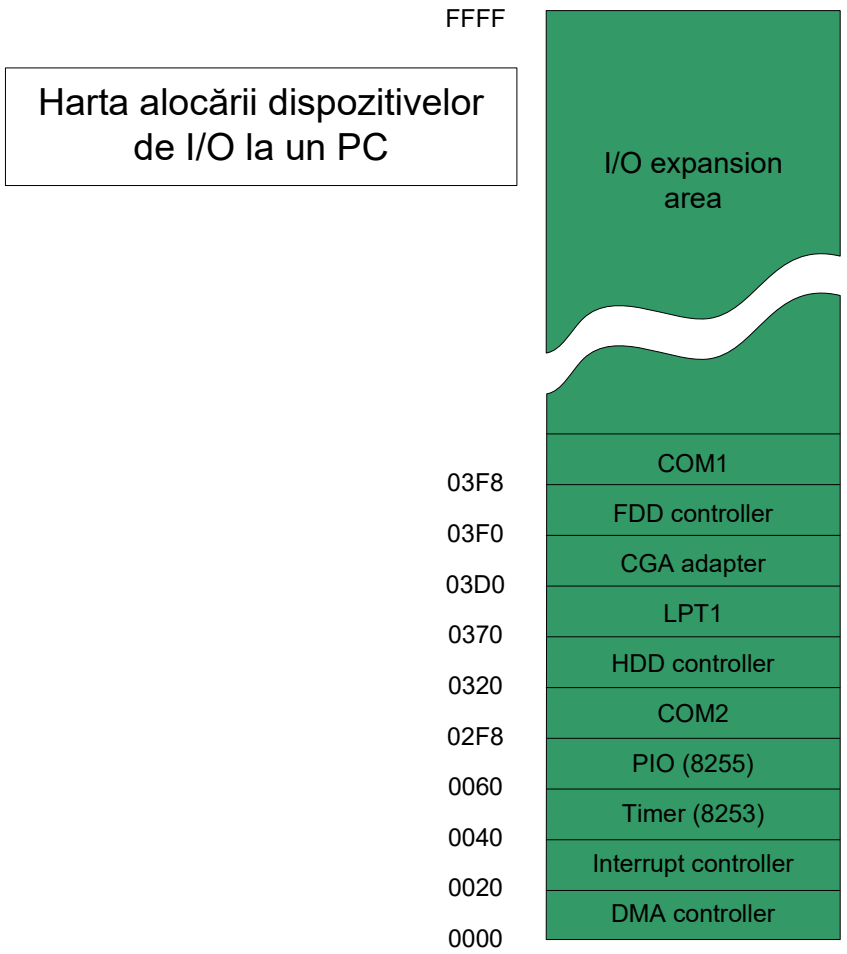
Structura şi organizarea calculatoarelor

- Memoria este împărțită la nivel logic în *pagini* de câte 64kB

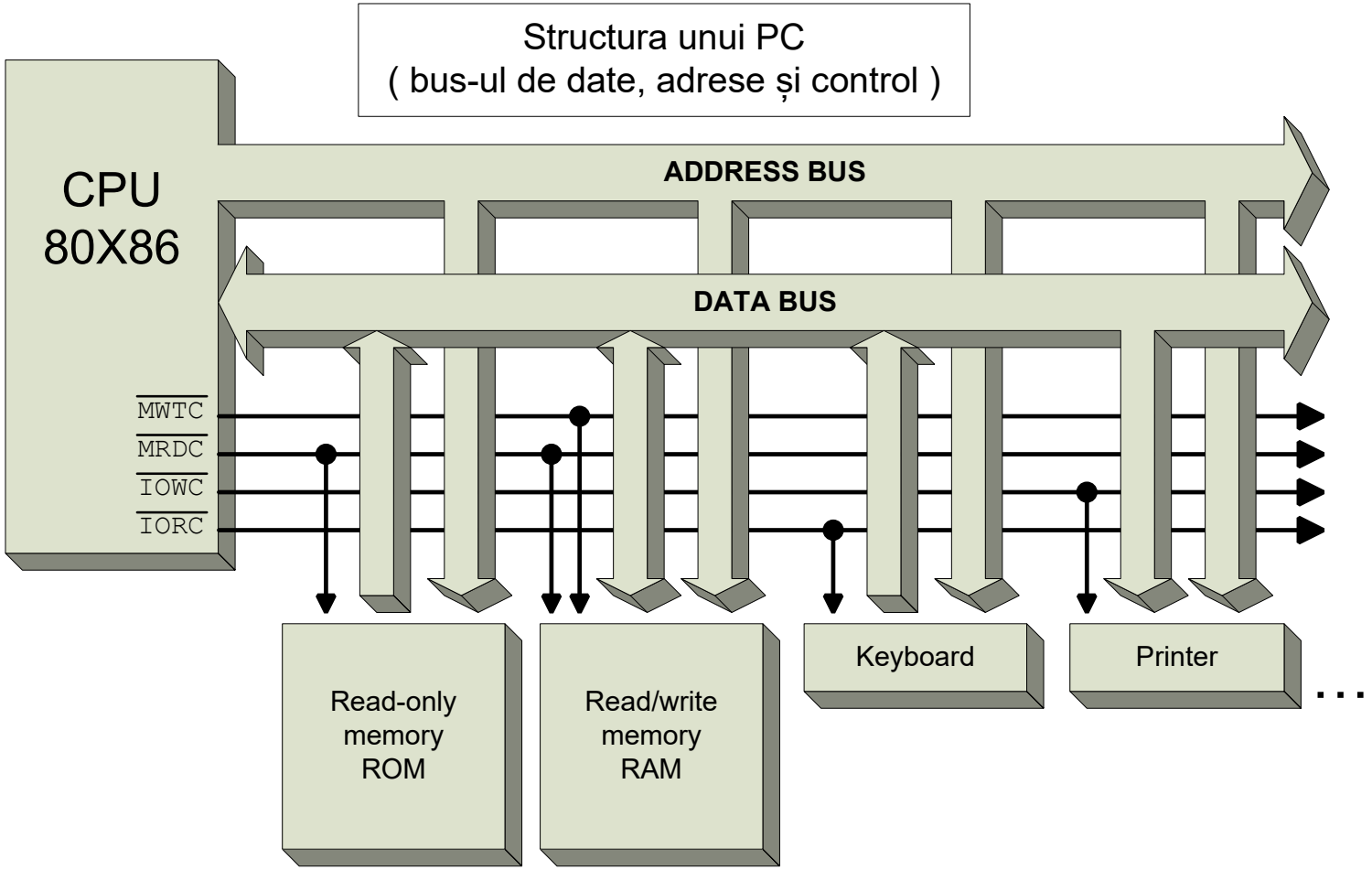


Structura şi organizarea calculatoarelor

- Fiecare periferic dispune de registre de control și registre de stare
- Aceste registre se regăsesc începând cu adresa indicată în figură



Structura şi organizarea calculatoarelor



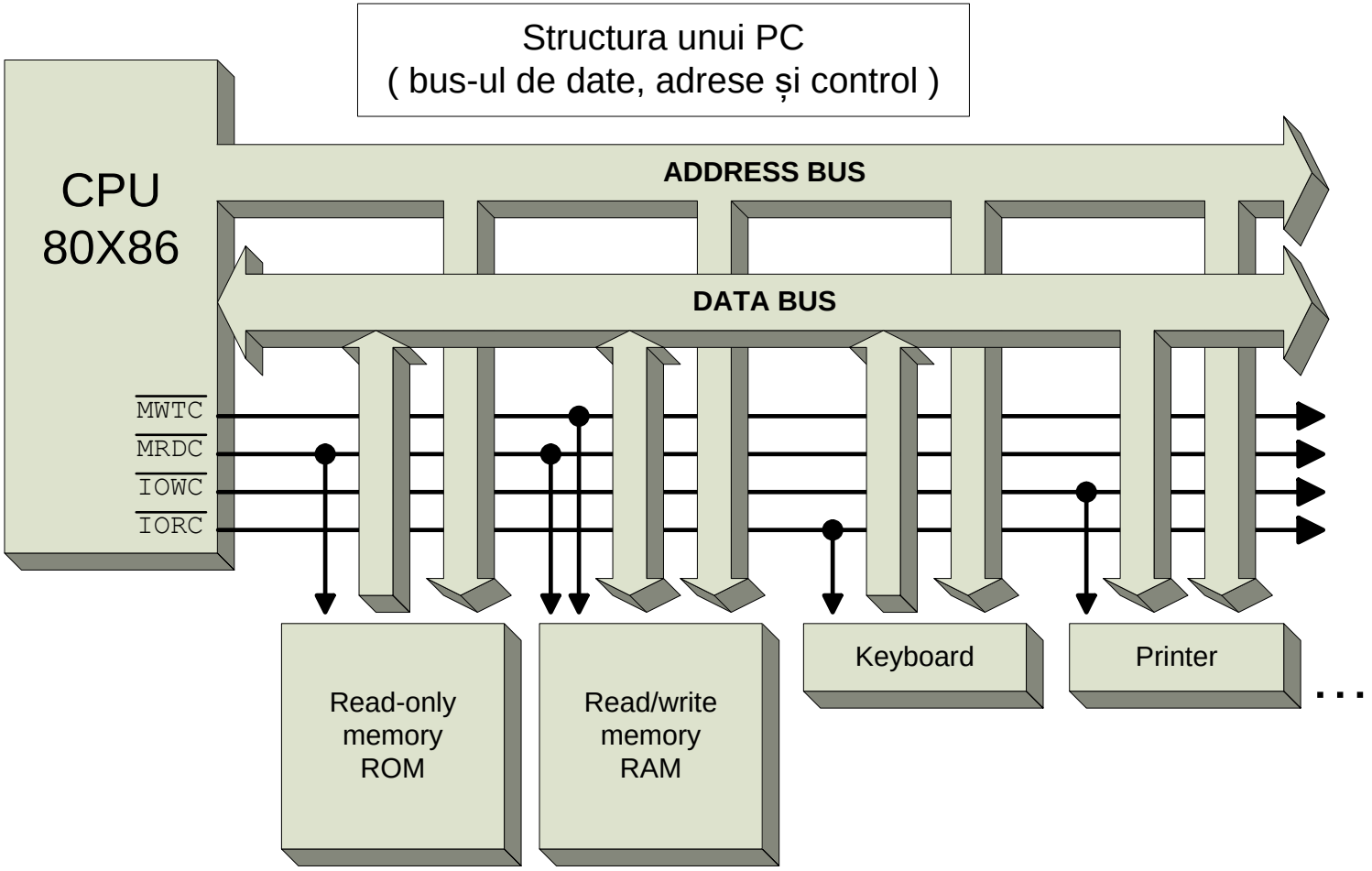
Activitate pentru laborator

- microprocesoare CISC sau RISC

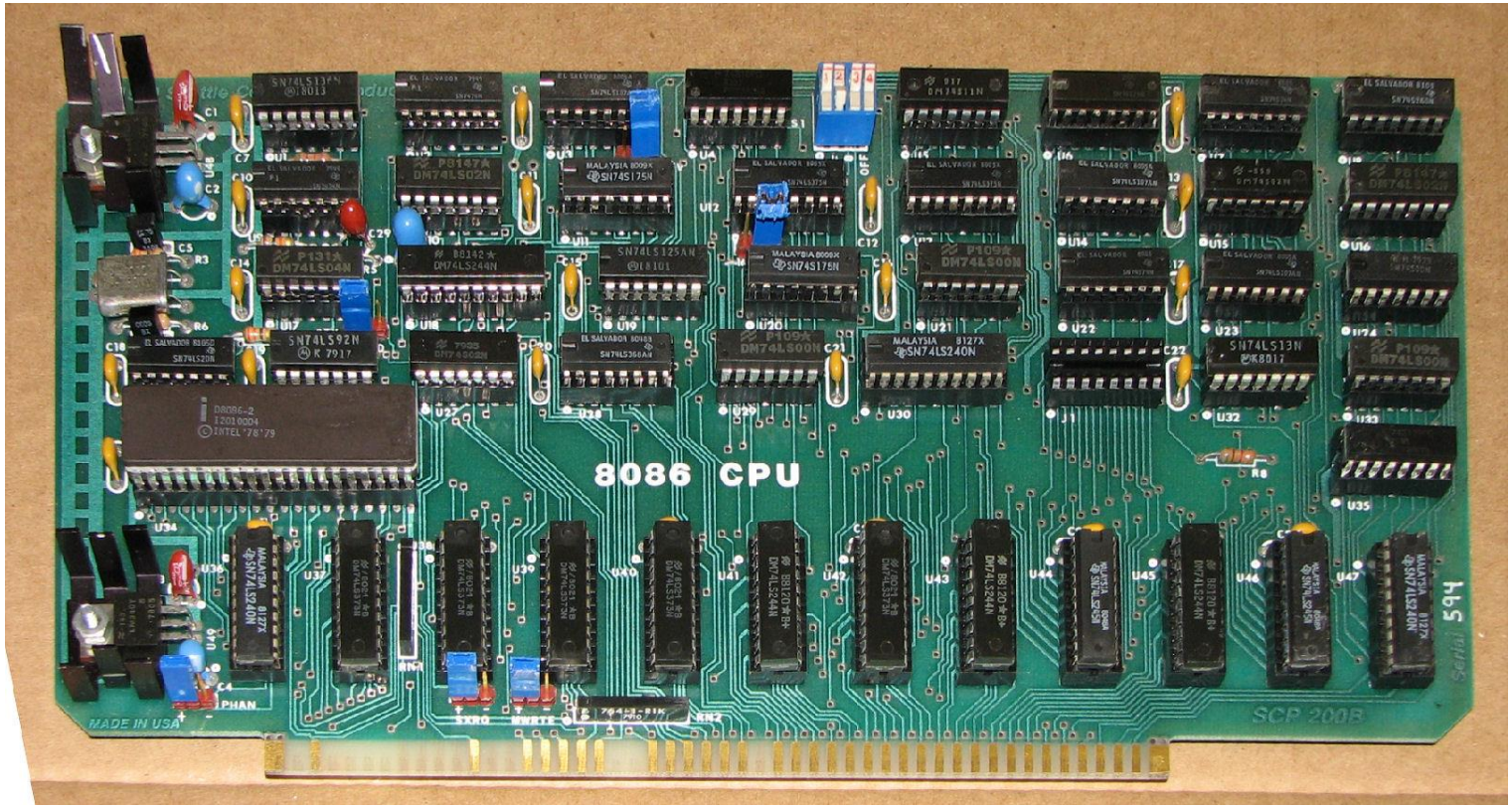
studiu de caz: microcontrolerul ATmega16 (RISC)

- **memoria**
- **periferice**
- **magistrala**

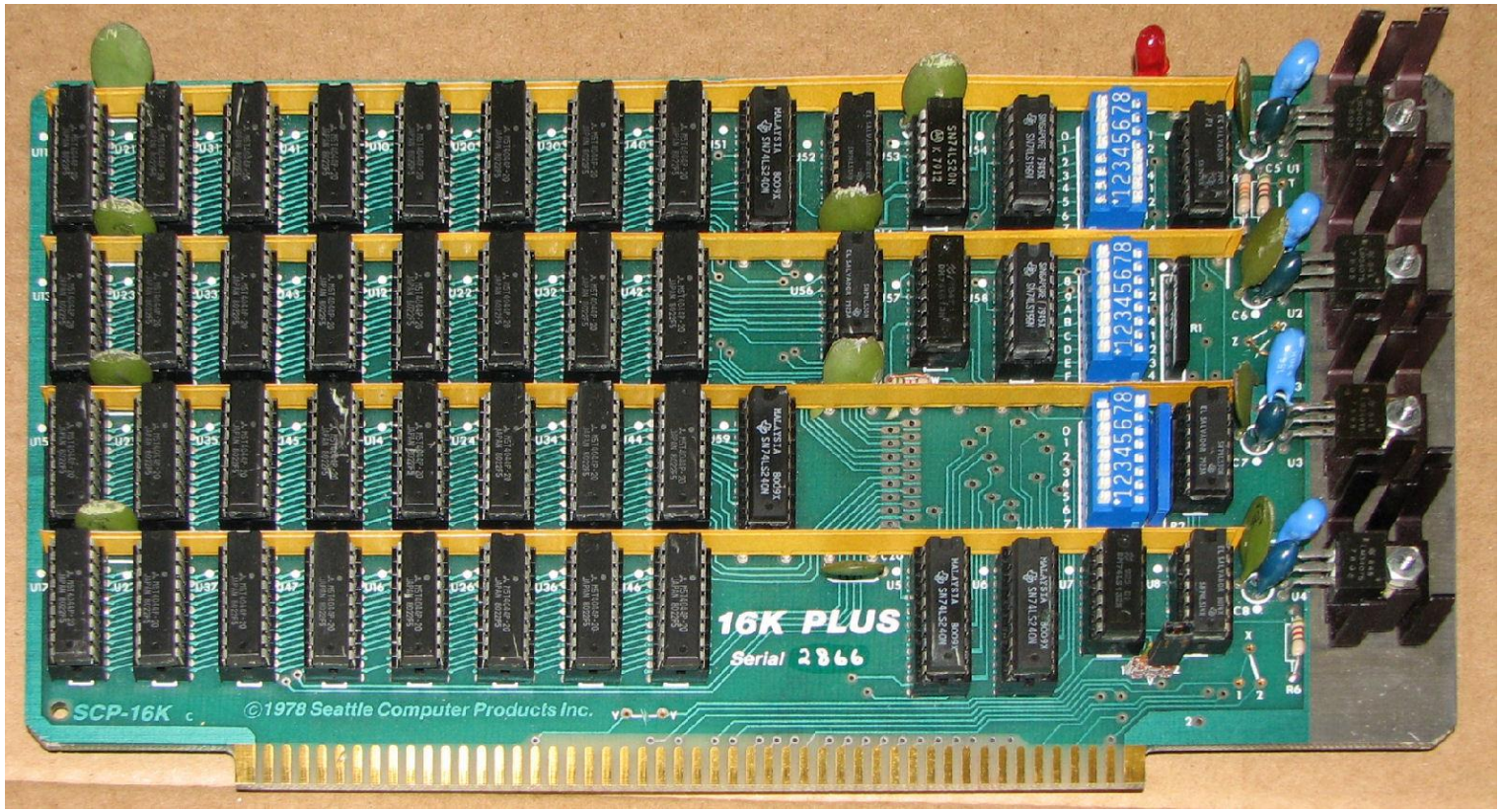
Structura şi organizarea calculatoarelor



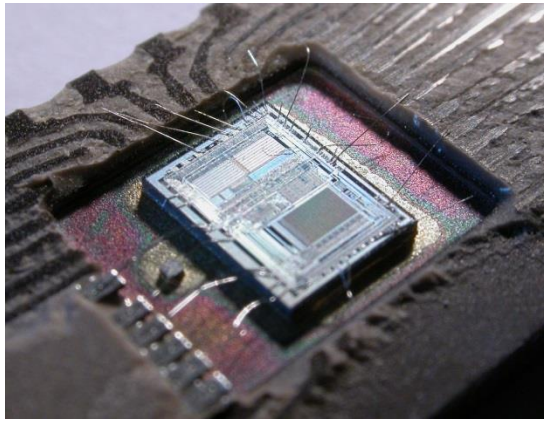
8086 – CPU speed 5MHz ☺



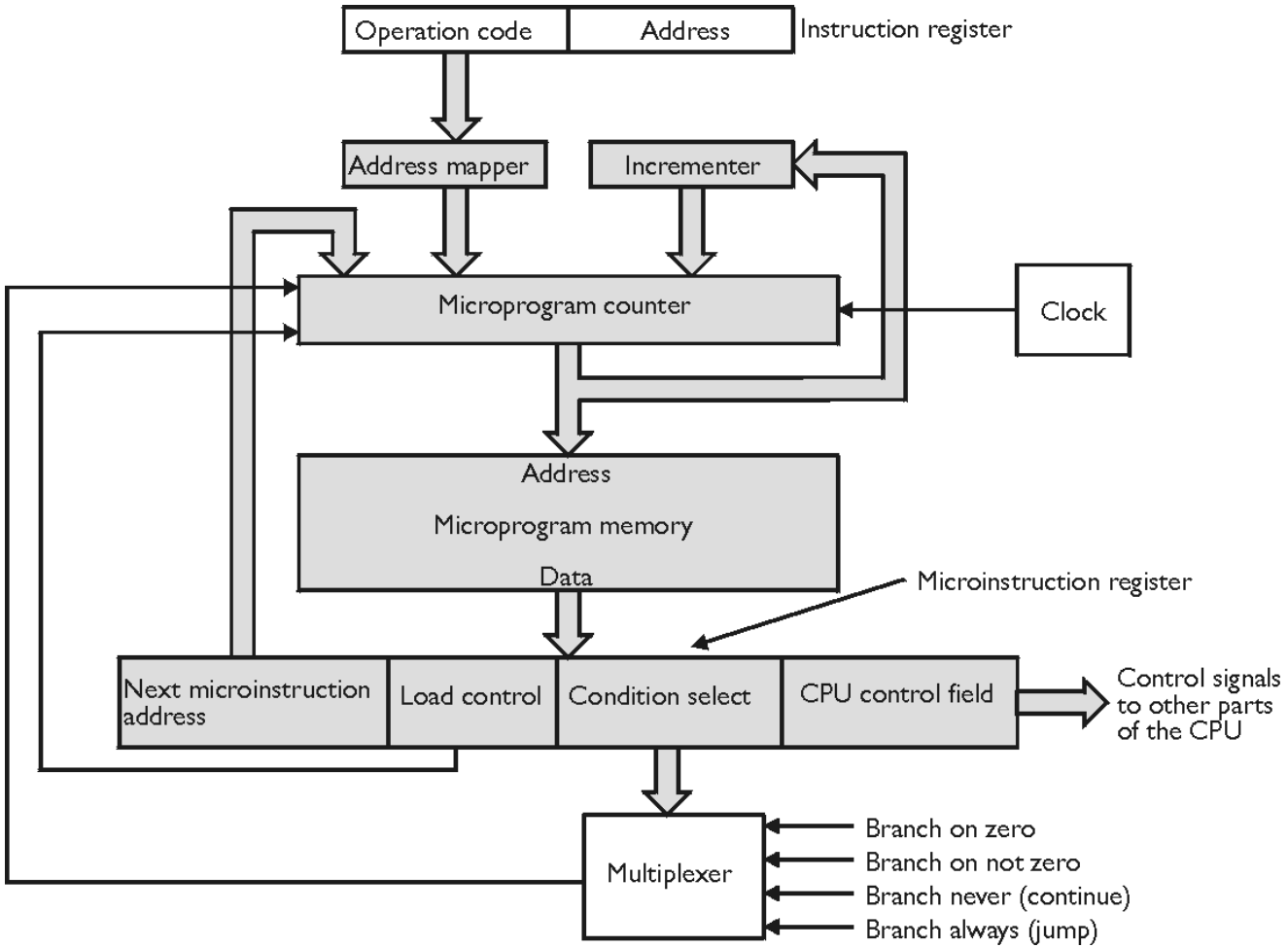
8086 – Memorie 16Ko ☺



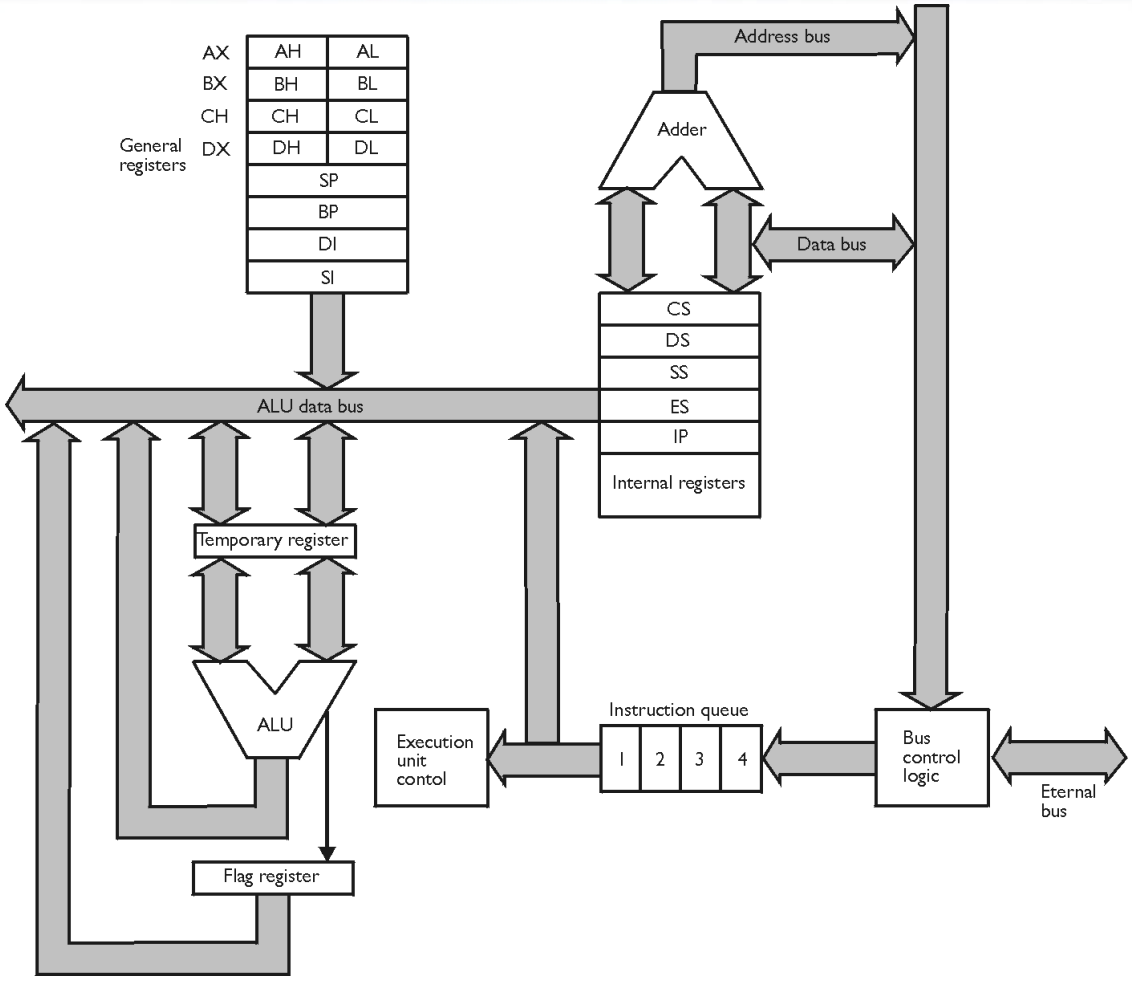
8086 – Structura internă ☺



Structura microprogramată

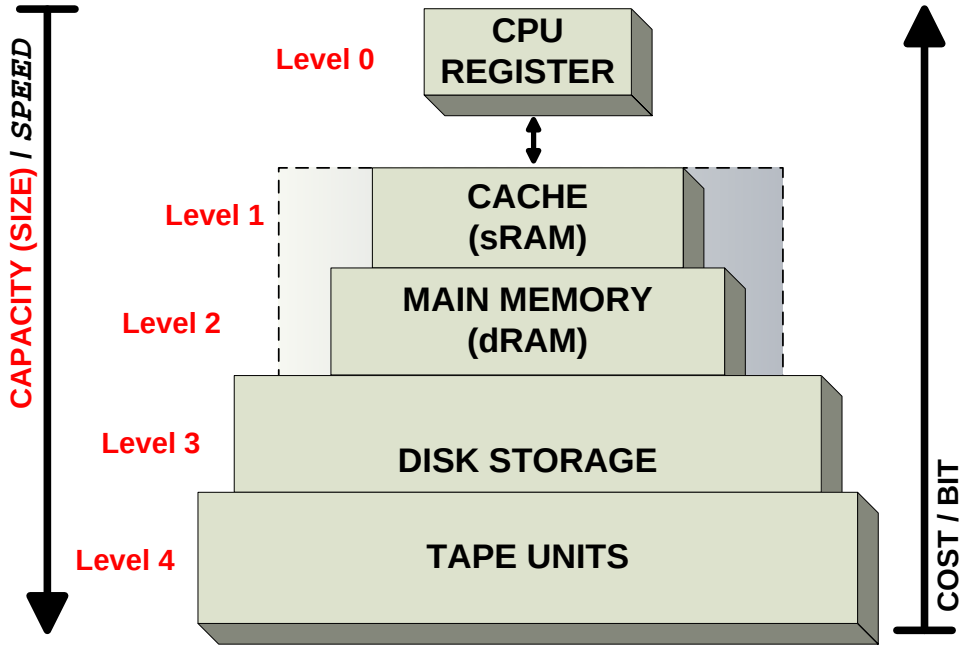


8086 – Structura internă



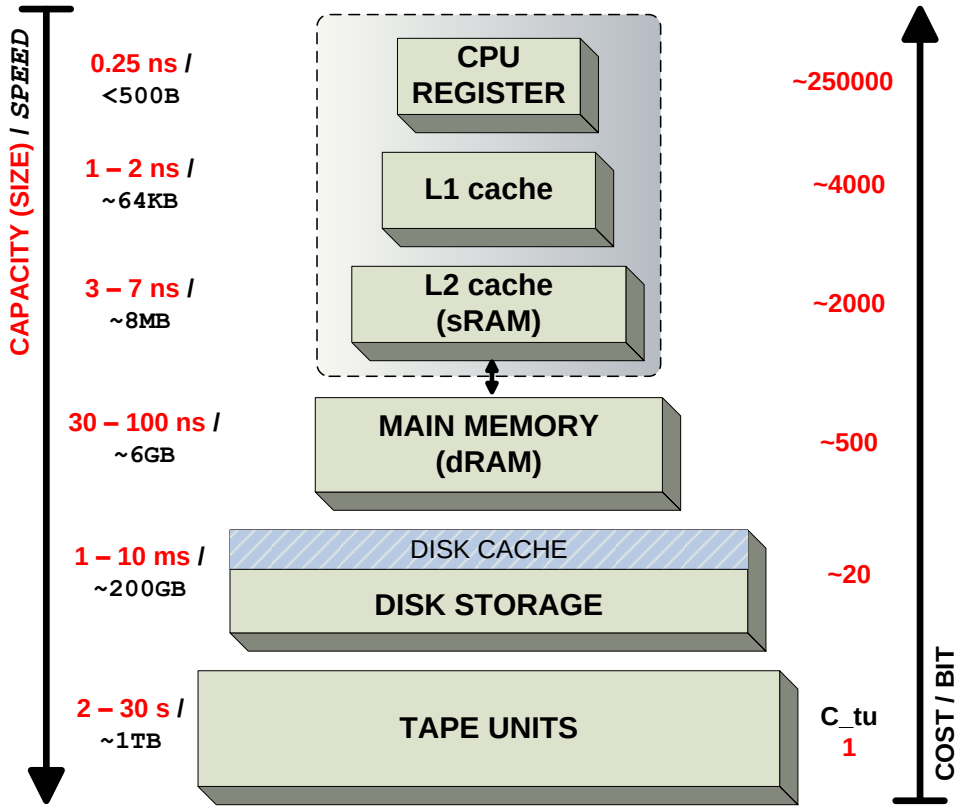
Organizarea memoriei

Clasificarea memoriei
unui sistem de calcul

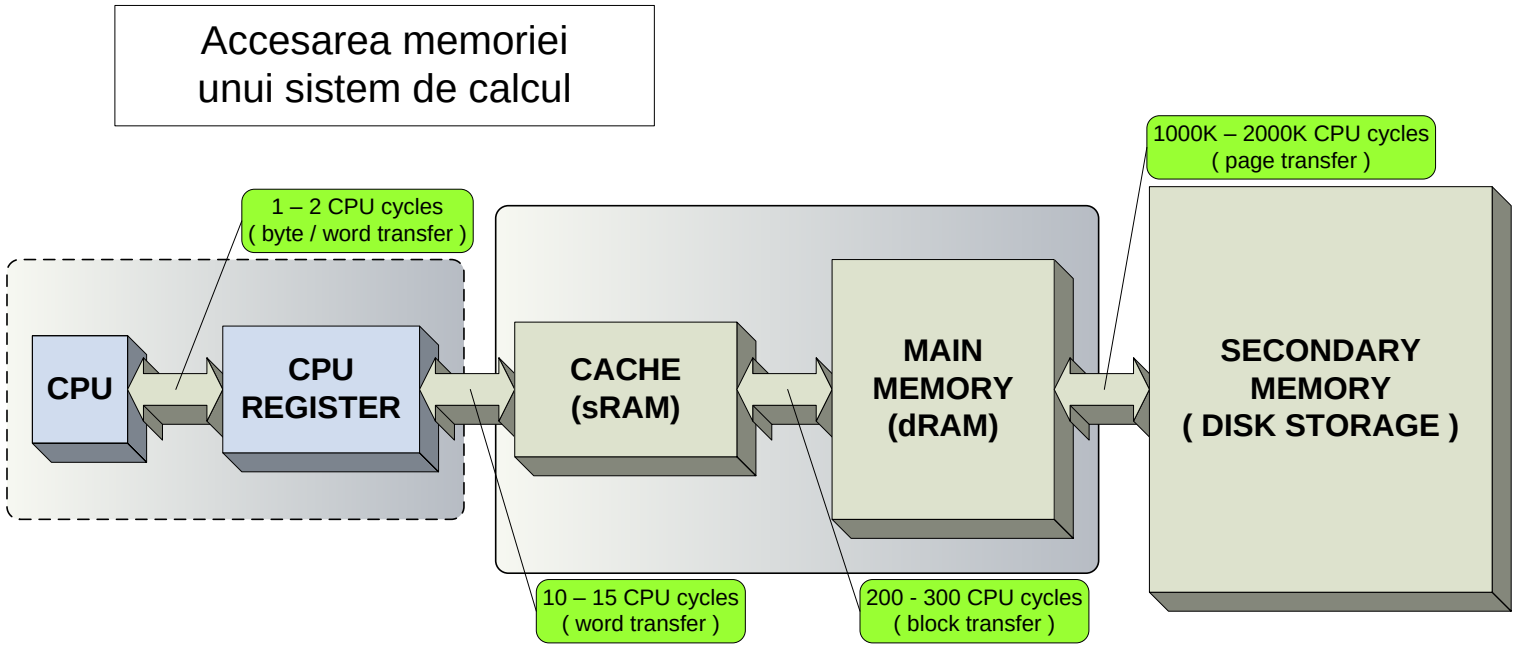


Organizarea memoriei

Clasificarea memoriei unui sistem de calcul



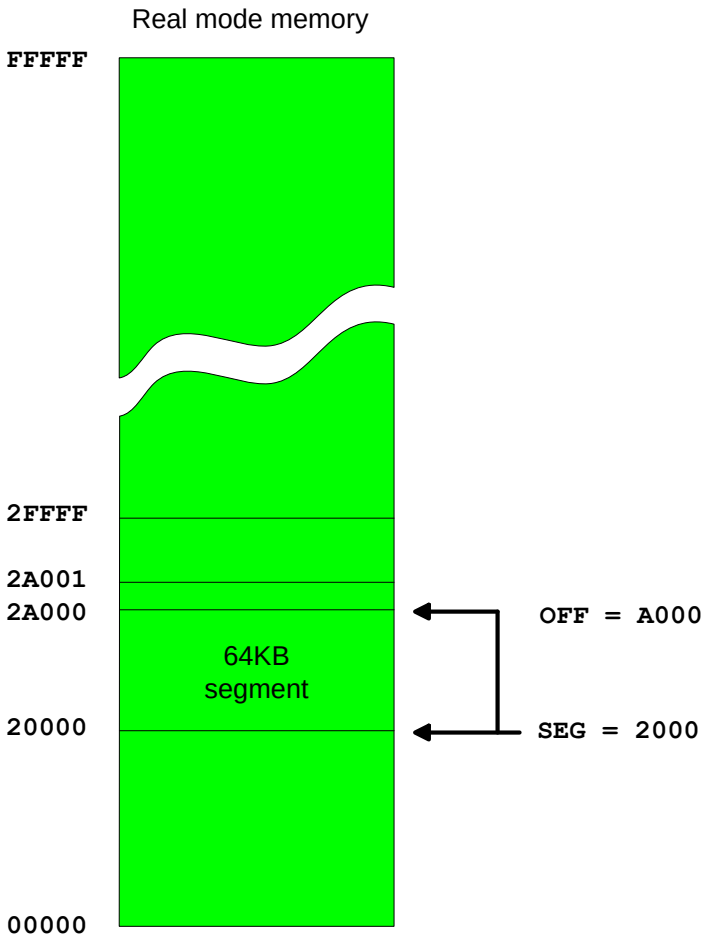
Accesarea memoriei



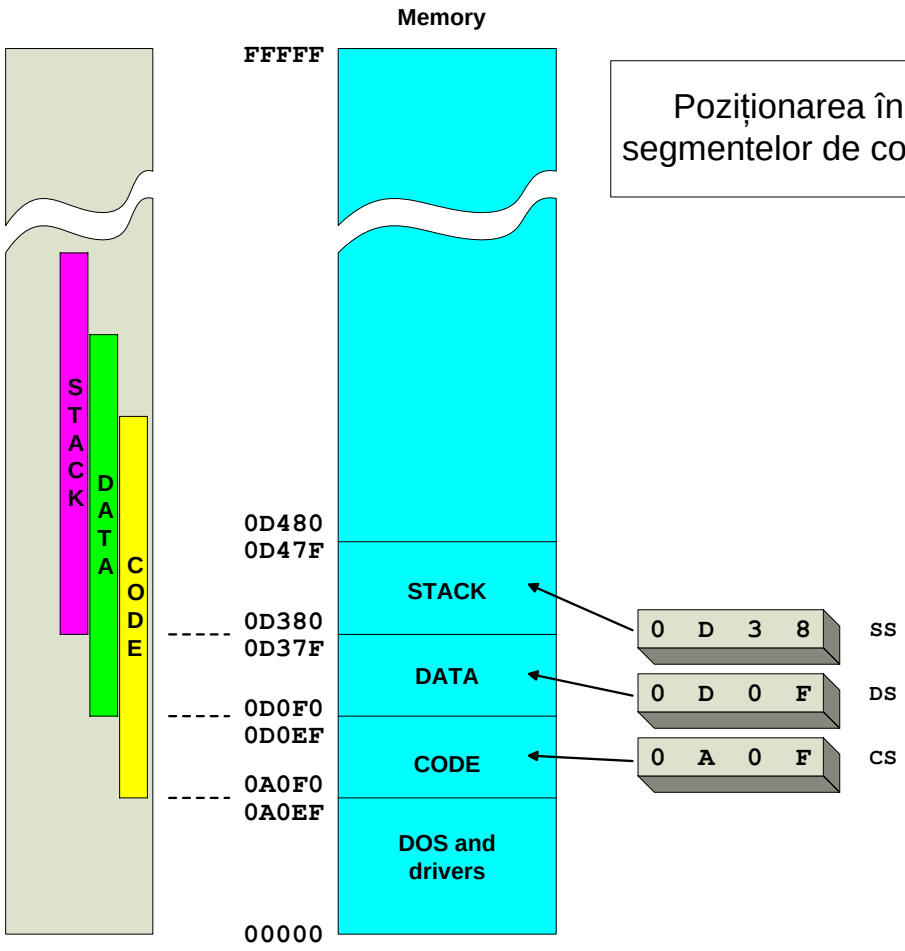
8086 – Structura memoriei

Adresarea în mod real 80x86
(memoria convențională)

Calcul adresă fizică (A.F.)
A.F. = SEG << 4 + OFF

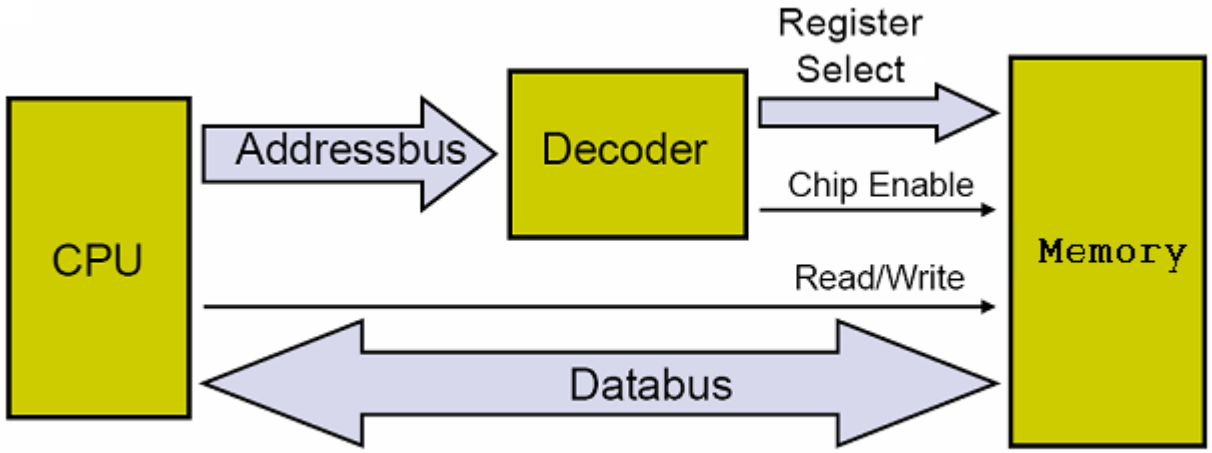


8086 – Structura memoriei

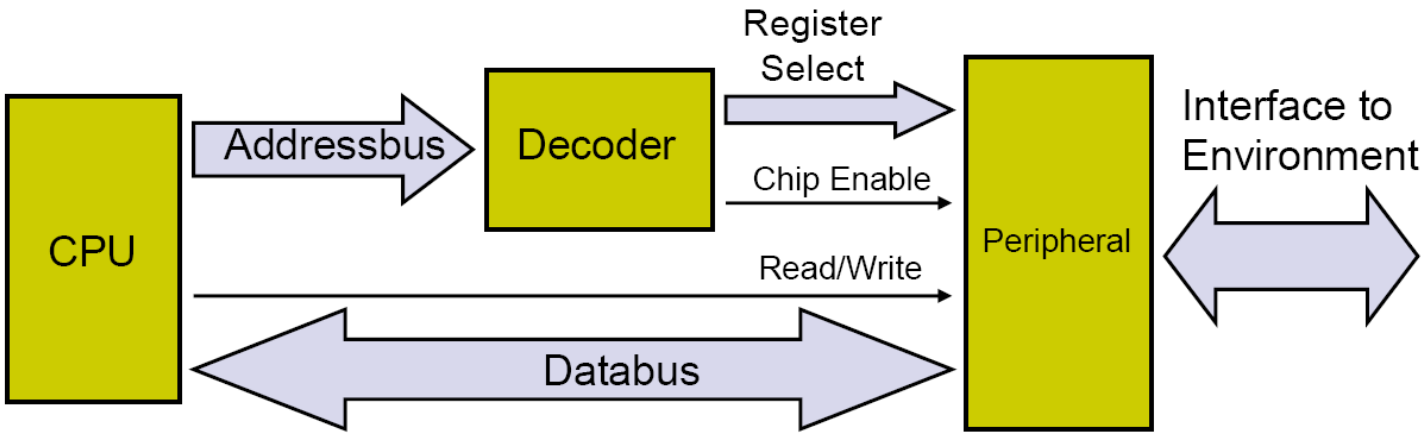


Poziționarea în memorie a segmentelor de cod, date și stiva

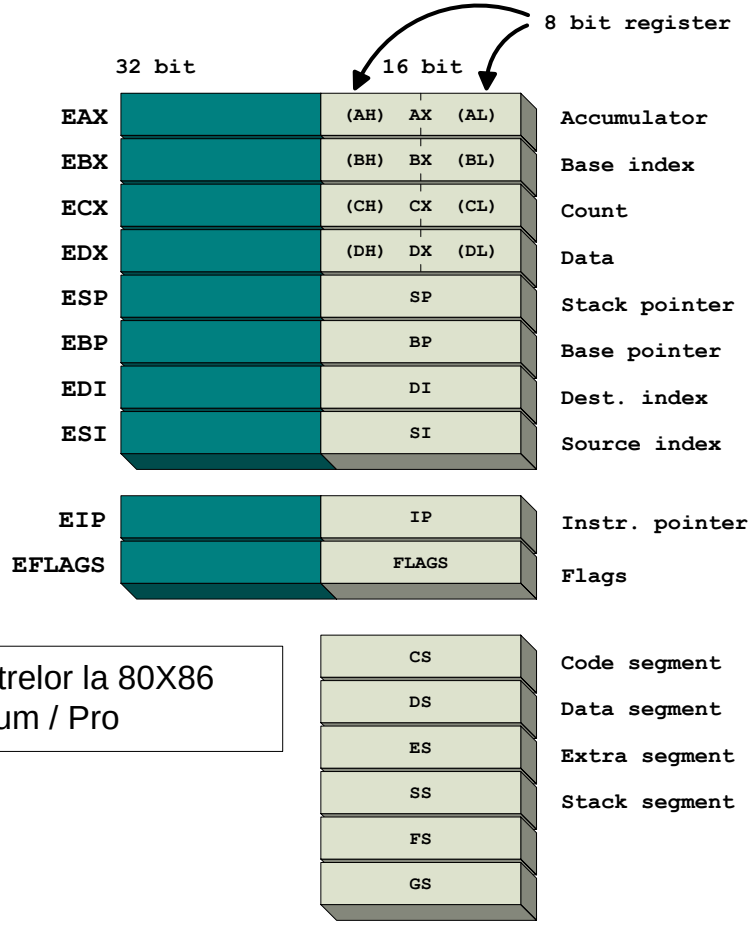
Maparea memoriei – general



Maparea I/O – general



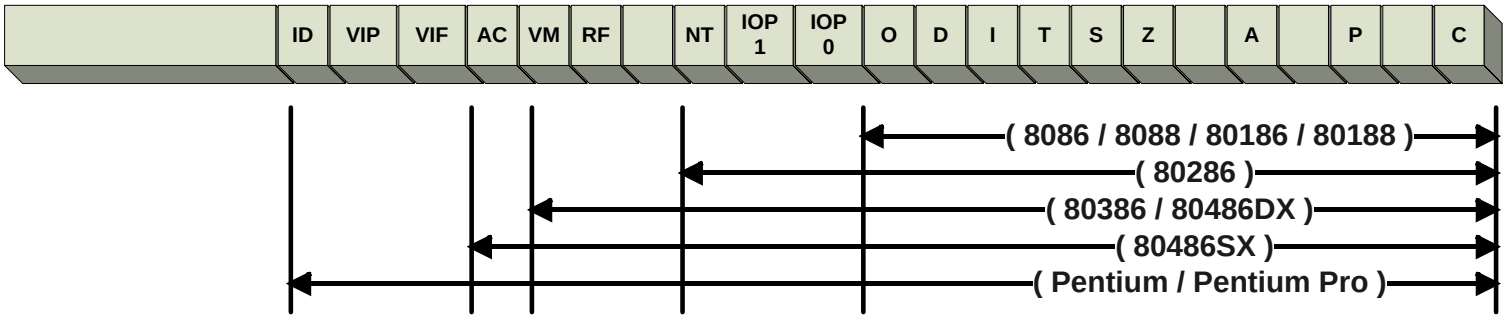
80x86 – Registre



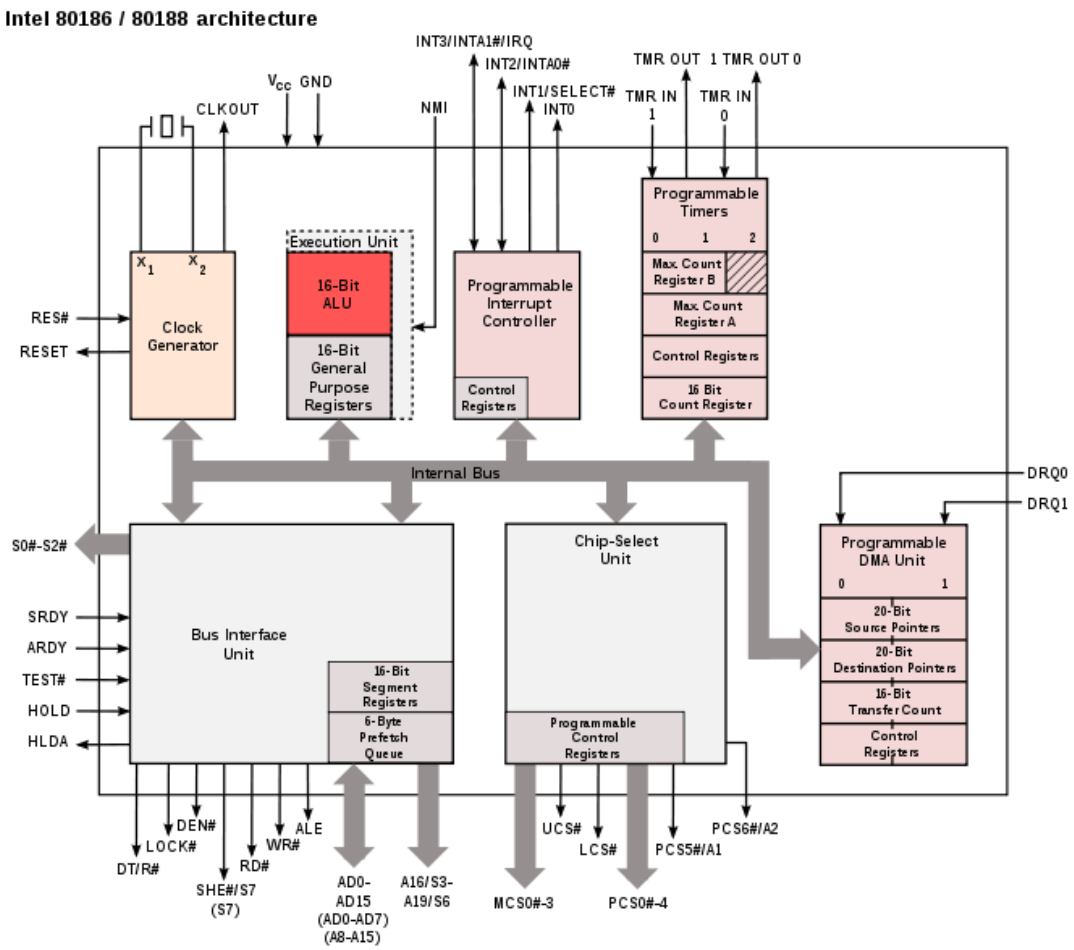
Structura registrelor la 80X86
si Pentium / Pro

80x86 – Registrele EFLAG și FLAG

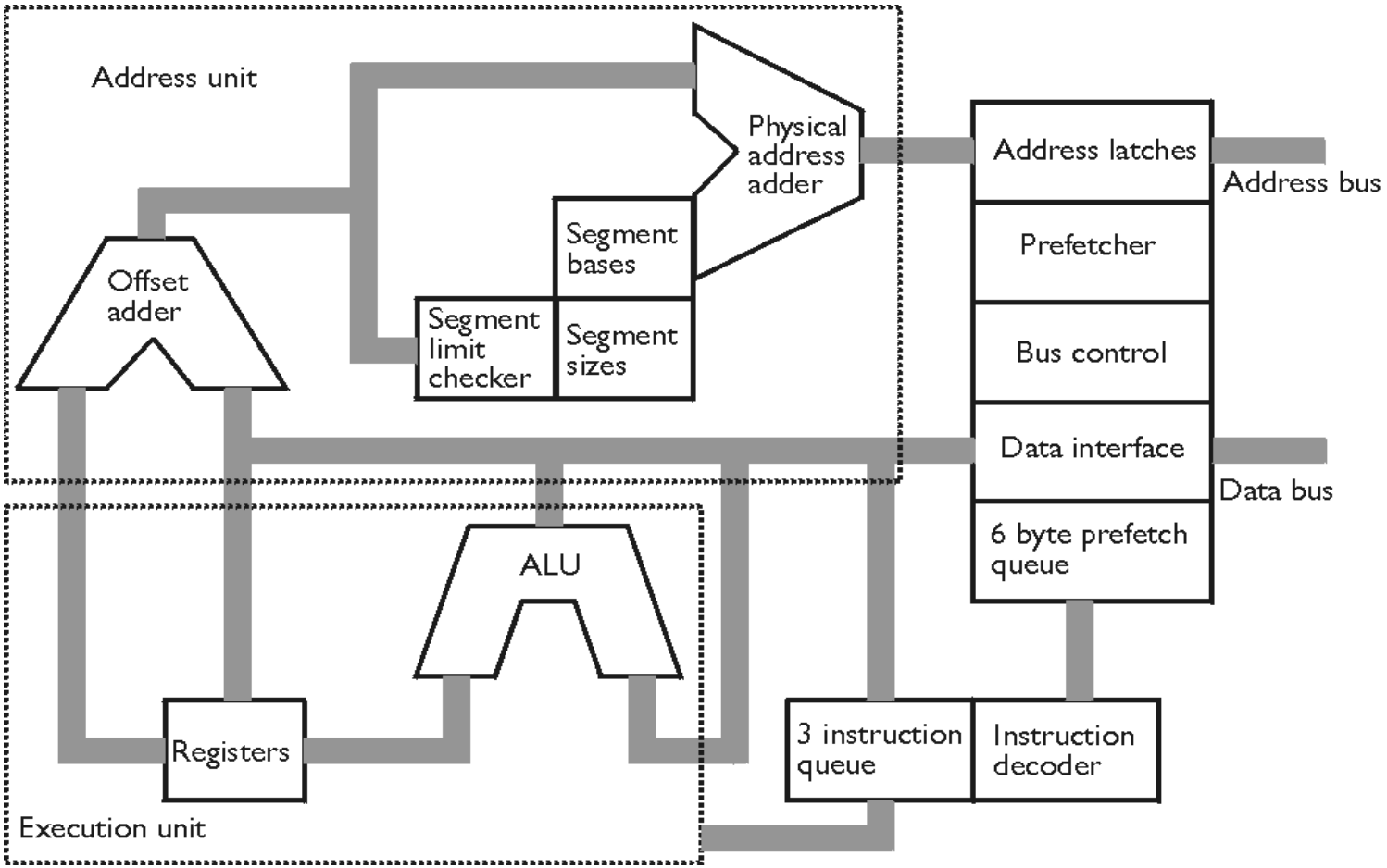
Structura registrului EFLAG și FLAG
la 80X86 si Pentium / Pro



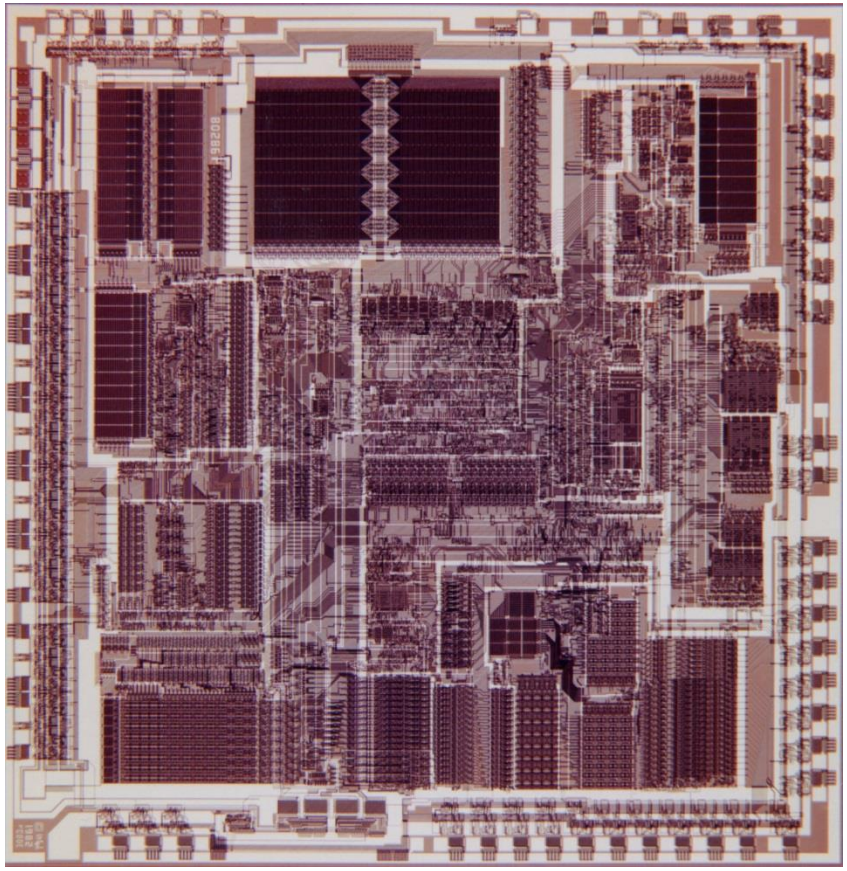
80186 – 80188


Intel 80186 / 80188 architecture
INT3/INTA1#/IRQ
INT2/INTA0#
INT1/SELECT#
INT0
NMI
TMR OUT 1
TMR OUT 0
TMR IN 1
TMR IN 0
CLKOUT
Vcc
GND
Programmable Timers
0 1 2
Max. Count Register B
Max. Count Register A
Control Registers
16 Bit Count Register
Execution Unit
16-Bit ALU
16-Bit General Purpose Registers
Programmable Interrupt Controller
Control Registers
X1
X2
Clock Generator
RES#
RESET
Internal Bus
DRQ0
DRQ1
Programmable DMA Unit
0 1
20-Bit Source Pointers
20-Bit Destination Pointers
16-Bit Transfer Count
Control Registers
Chip-Select Unit
Programmable Control Registers
S0#-S2#
SRDY
ARDY
TEST#
HOLD
HLDA
Bus Interface Unit
16-Bit Segment Registers
6-Byte Prefetch Queue
DEN#
ALE
LOCK#
WR#
DT/R#
RD#
SHE#/S7 (S7)
AD0-AD15 (AD0-AD7) (A8-A15)
A16/S3-A19/S6
UCS#
LCS#
PCS5#/A1
PCS6#/A2
MCS0#-3
PCS0#-4

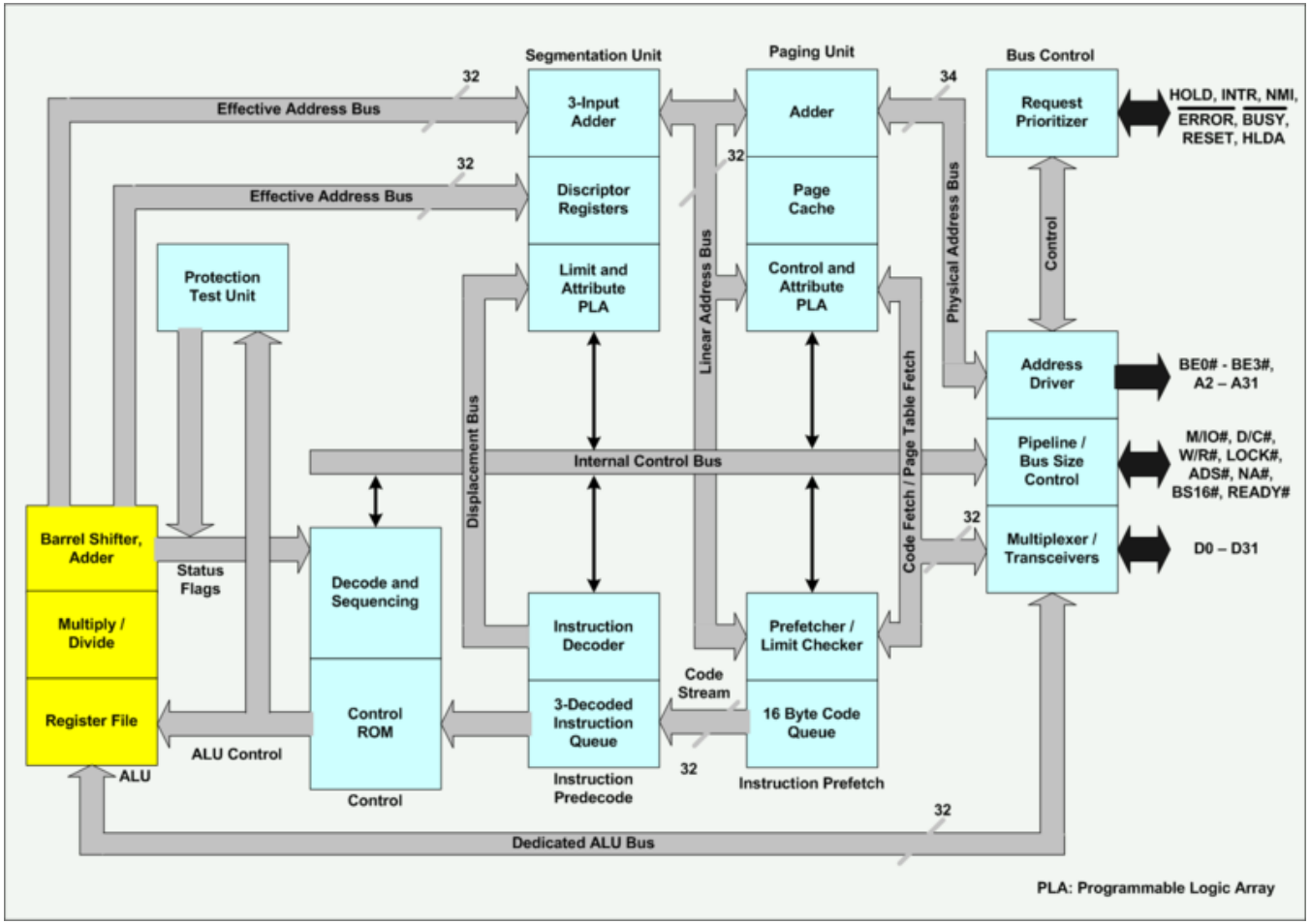
80286 – 16 Mb – real & protejat



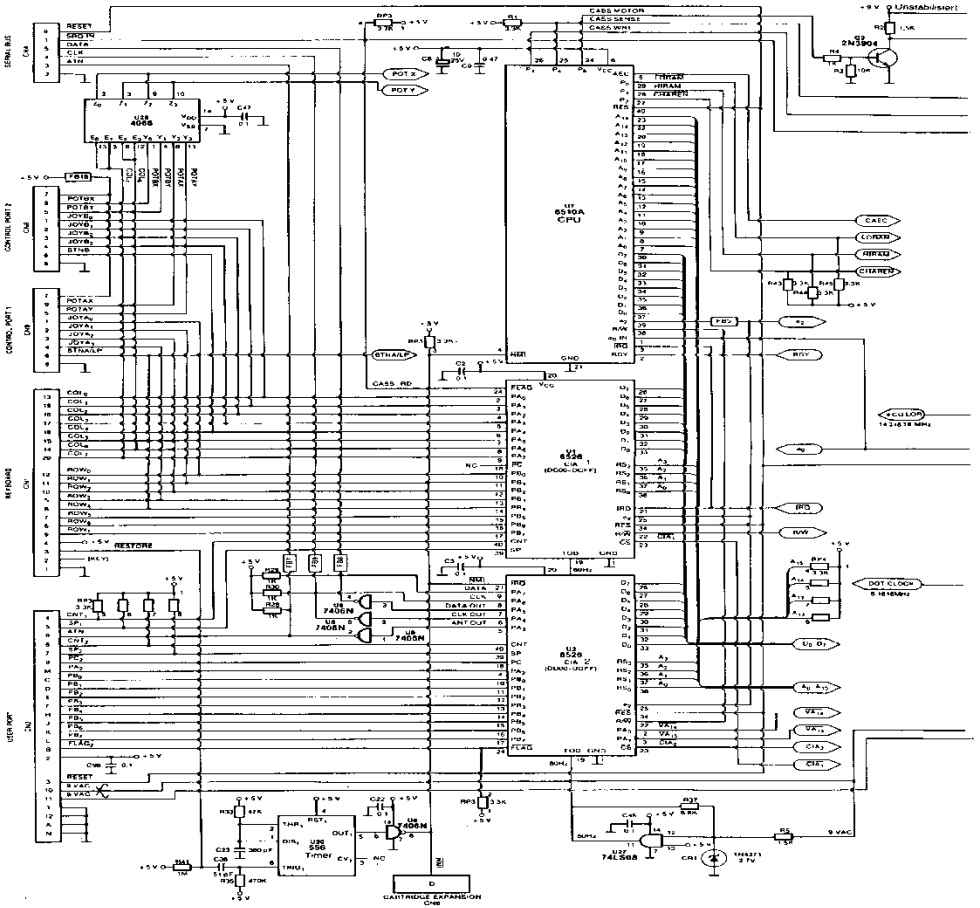
80286 – Structura internă



80386DX


Segmentation Unit
Paging Unit
Bus Control
32
Effective Address Bus
3-Input Adder
Adder
34
Request Prioritizer
HOLD, INTR, NMI,
ERROR, BUSY,
RESET, HLDA
32
Effective Address Bus
Discriptor Registers
32
Page Cache
Protection Test Unit
Limit and Attribute PLA
Control and Attribute PLA
Linear Address Bus
Physical Address Bus
Control
Address Driver
BE0# - BE3#,
A2 – A31
Displacement Bus
Internal Control Bus
Code Fetch / Page Table Fetch
Pipeline / Bus Size Control
M/IO#, D/C#,
W/R#, LOCK#,
ADS#, NA#,
BS16#, READY#
Barrel Shifter, Adder
Status Flags
Decode and Sequencing
32
Multiplexer / Transceivers
D0 – D31
Multiply / Divide
Instruction Decoder
Prefetcher / Limit Checker
Code Stream
Register File
Control ROM
3-Decoded Instruction Queue
16 Byte Code Queue
ALU Control
ALU
Control
Instruction Predecode
32
Instruction Prefetch
32
Dedicated ALU Bus
PLA: Programmable Logic Array

Activitate pentru laborator ☺



Structura şi organizarea calculatoarelor

- Arhitecturi de procesoare
  - **80X86**
  - **ARM**
  - **PowerPC**
  - **MIPS**
  - **SPARC**
  - **V850**
  - **etc.**

Tipuri de procesoare ARM

- Există o gamă largă de procesoare bazate pe acelaşi nucleu ARM

Application Processors

ARM Cortex-A8
ARM Cortex-A9 MPCore
ARM Cortex-A9 Single Core Processor
ARM11 MPCore
ARM1136J(F)-S
ARM1176JZ(F)-S
ARM720T
ARM920T
ARM922T
ARM926EJ-S

Embedded Processors

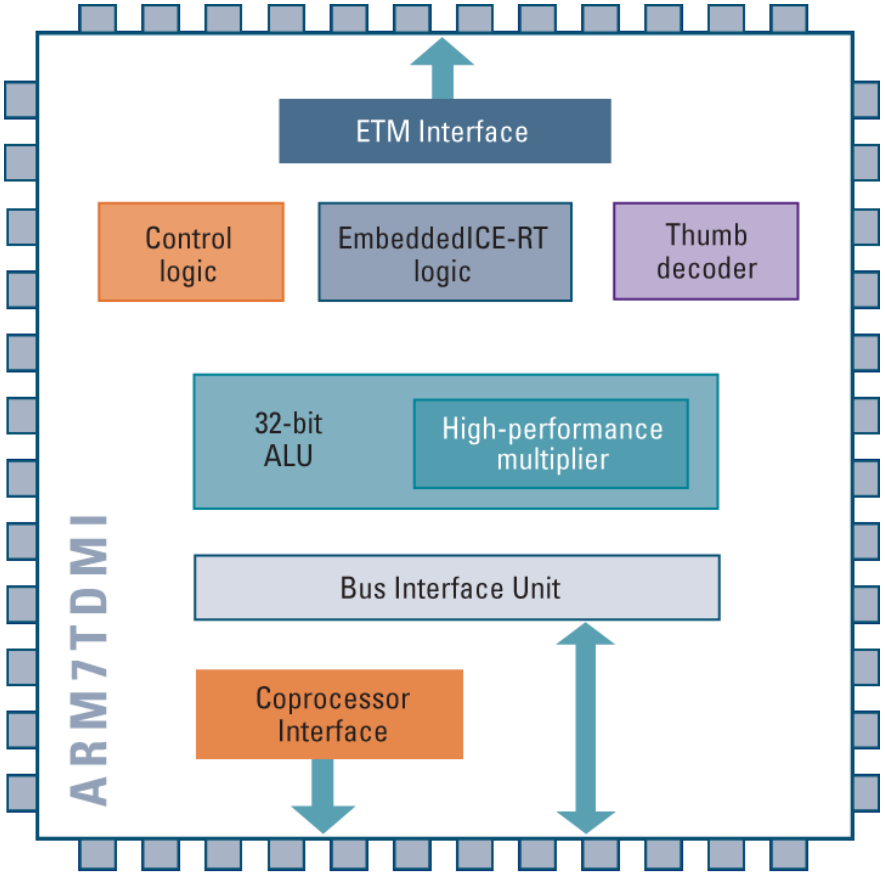
ARM Cortex-M0
ARM Cortex-M1
ARM Cortex-M3
ARM Cortex-R4(F)
ARM1156T2(F)-S
ARM7EJ-S
ARM7TDMI
ARM7TDMI-S
ARM946E-S
ARM966E-S
ARM968E-S

Secure Cores

SC300
SecurCore SC100
SecurCore SC200

ARM7TDMI

The ARM7TDMI core is a 32-bit embedded RISC processor delivered as a hard macro cell optimized to provide the best combination of performance, power and area characteristics. The ARM7TDMI core enables system designers to build embedded devices requiring small size, low power and high performance.



ARM 32-bit RISC

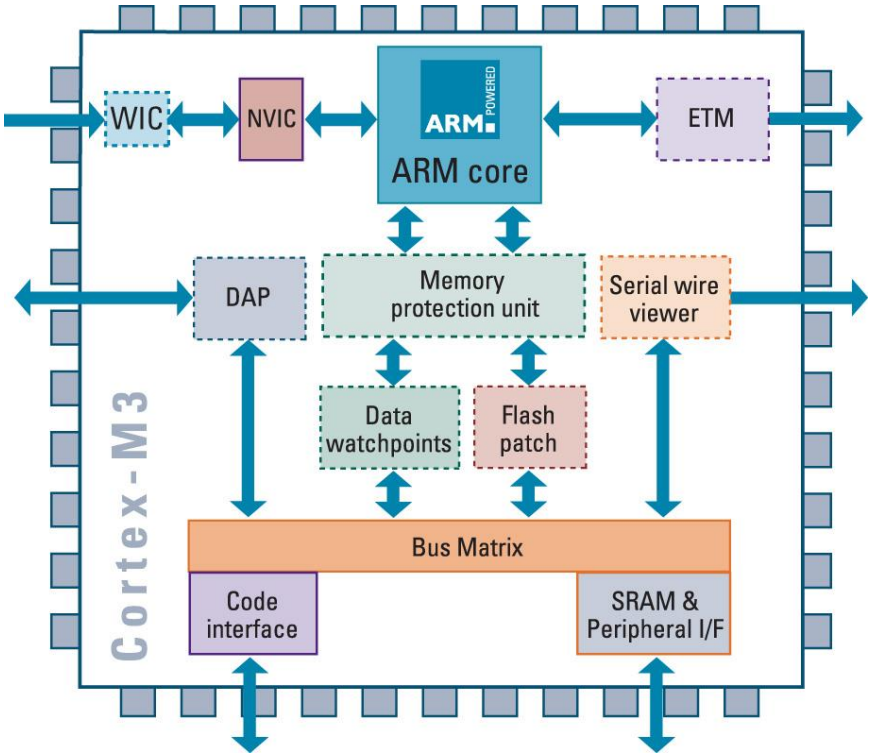
- Applications:
  - **Personal audio (MP3, WMA, AAC players)**
  - **Entry level wireless handset**
  - **Pager**
  - **Ink-jet/bubble-jet printer**
  - **Digital still camera**

- Features:
  - **32/16-bit RISC architecture (ARM v4T)**
  - **32-bit ARM instruction set for maximum performance and flexibility**
  - **16-bit Thumb instruction set for increased code density**
  - **Unified bus interface, 32-bit data bus carries both instructions and data**
  - **Three-stage pipeline**
  - **32-bit ALU**
  - **Very small die size and low power consumption**
  - **Coprocessor interface**

- Extensive debug facilities:
  - **Embedded ICE-RT real-time debug unit**
  - **JTAG interface unit**
  - **Interface for direct connection to Embedded Trace Macro cell (ETM)**

ARM Cortex-M3

- The central CM3Core is based on the Harvard architecture characterized by separate buses for instructions and data
- Embedded Trace Macrocells (ETM) provide comprehensive debug and trace facilities for ARM processors.



ARM Cortex-M3 – Tipuri de aplicații

The Cortex-M3 processor offers an excellent balance of architectural features, high performance and low costs, making it a very attractive choice for a broad range of applications, including:

- **Microcontrollers**
  - 32-bit performance at 8-bit costs
- **Wireless networking (inc Bluetooth, ZigBee and others)**
  - Low power operation and integrated sleep modes supporting complex stacks
- **Automotive and industrial control systems**
  - Secure, reliable and deterministic operation
- **White goods**
  - High performance maths for complex motor algorithm support
- **Electronic toys**
  - Low cost implementations for next generation intelligent toys
- **Medical instrumentation**
  - High reliability core and tools enabling IEC61508 and FDA approval.

ARM Cortex-M3 – Caracteristici

- ARMv7-M architecture
  - **Optimized for microcontroller and low-cost applications**
- Thumb-2 instruction set
  - **Enhanced levels of performance, energy efficiency, and code density**
  - **Mixed mode capability implies no need to interwork between modes**
  - **ARM levels of performance with Thumb level code density**
- Single cycle multiply and hardware divide instructions
  - **32-bit multiplication in a single cycle**
  - **Signed and unsigned divide operations between 2 and 12 cycles**
- Preconfigured memory map
- Up to 4 gigabytes of addressable memory space
- Predefined addresses for code, memory, external devices, peripherals
- Dedicated space for vendor specific addressability

ARM Cortex-M3 – Caracteristici

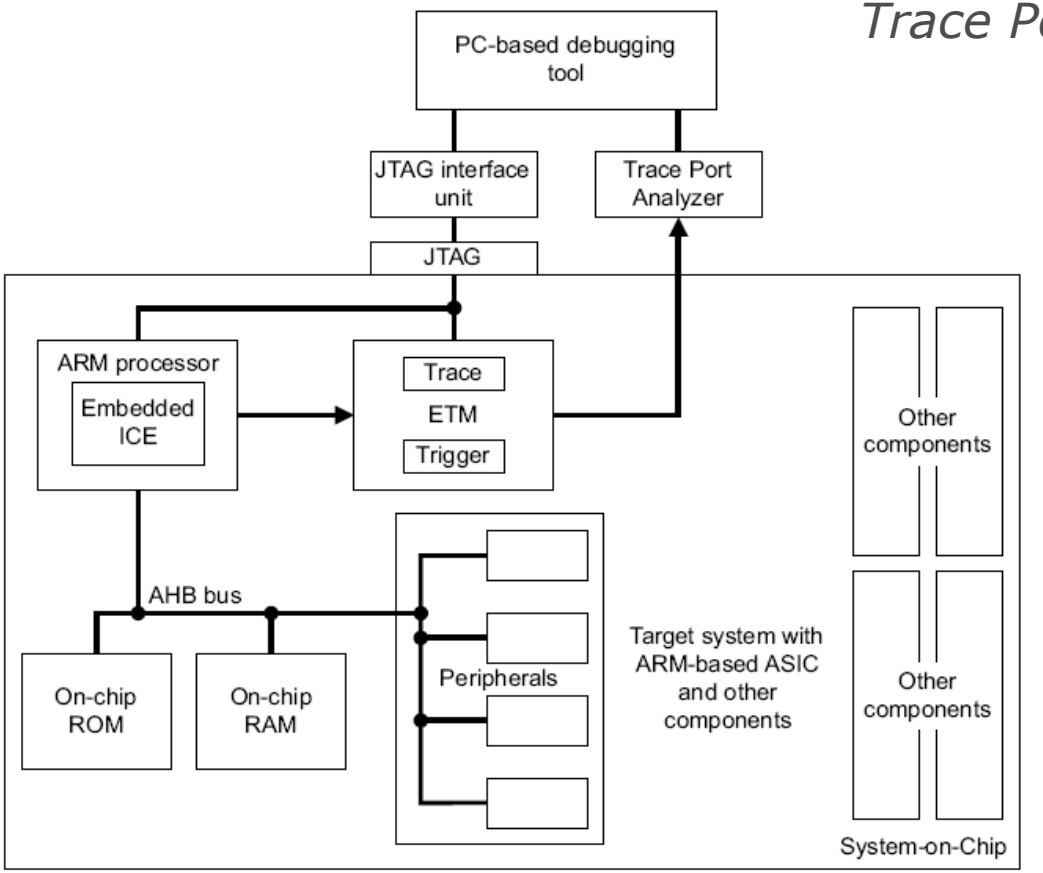
- **Hierarchical structure with tightly integrated peripherals**
  - **CM3Core**
    - Harvard bus architecture – separate instruction and data buses
    - Highly efficient 3-stage pipeline with branch speculation
    - Nested Vectored Interrupt Controller (NVIC)
    - Gate efficient stack-based register model
    - Configurable from 1-240 physical interrupts; up to 256 levels of priority
    - Non - Maskable Interrupt (NMI) enables critical interrupt capabilities
    - Low latency through tail chaining, late arrival service & stack pop pre-emption
    - Nesting (stacking) of interrupts
    - Dynamic interrupt reprioritization
  - **Memory Protection Unit (MPU)**
    - Optional component for separation of processing tasks and data protection
    - Up to 8 regions of protection; each of which can be divided into 8 sub-regions
    - Region sizes between 32 bytes to the entire 4 gigabytes of addressable memory
  - **Embedded Trace Macrocell (ETM)**
    - Optional component for real-time instruction trace
  - **Data Watchpoint and Trace unit (DWT)**
    - Implements hardware breakpoints and provides instruction execution statistics
  - **Flash Patch and Breakpoint unit (FPB)**
    - Implements 6 program breakpoints and 2 literal data fetch breakpoints
  - **Debug Port (SW-DP or SWJ-DP)**
    - Configurable debug access through Serial Wire or JTAG interface

ARM Cortex-M3 – Caracteristici

- **Atomic bit manipulation with bit banding**
  - **Direct access to single bits of data**
  - **Two 1MB bit banding regions for memory and peripherals mapping to 32MB alias regions**
  - **Atomic operation, cannot be interrupted by other bus activities**
- **Unaligned data storage and access**
  - **Continuous storage of data requiring different byte lengths**
  - **Data access in a single core access cycle**
- **Integrated sleep modes**
  - **Sleep Now mode for immediate transfer to low power state**
  - **Sleep on Exit mode for entry into low power state after the servicing of an interrupt**
  - **Ability to extend power savings to other system components**

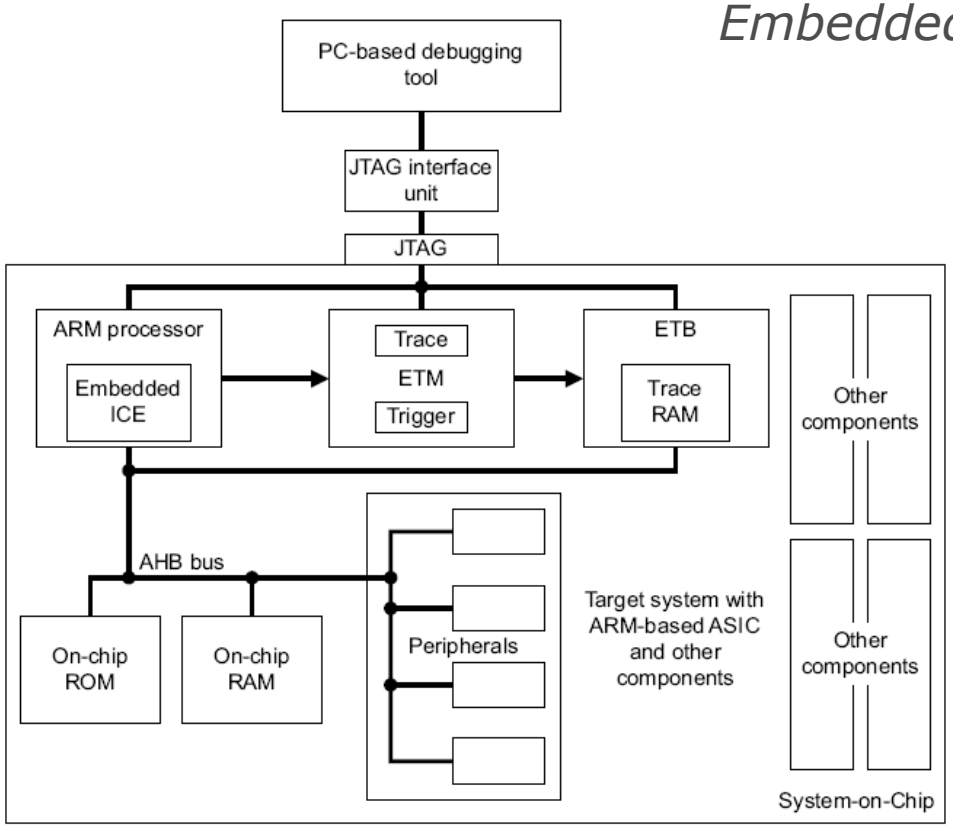
Debugging – TPA

Trace Port Analyzer (TPA)



Embedded Trace Macrocell™
ETMv1.0 to ETMv3.4

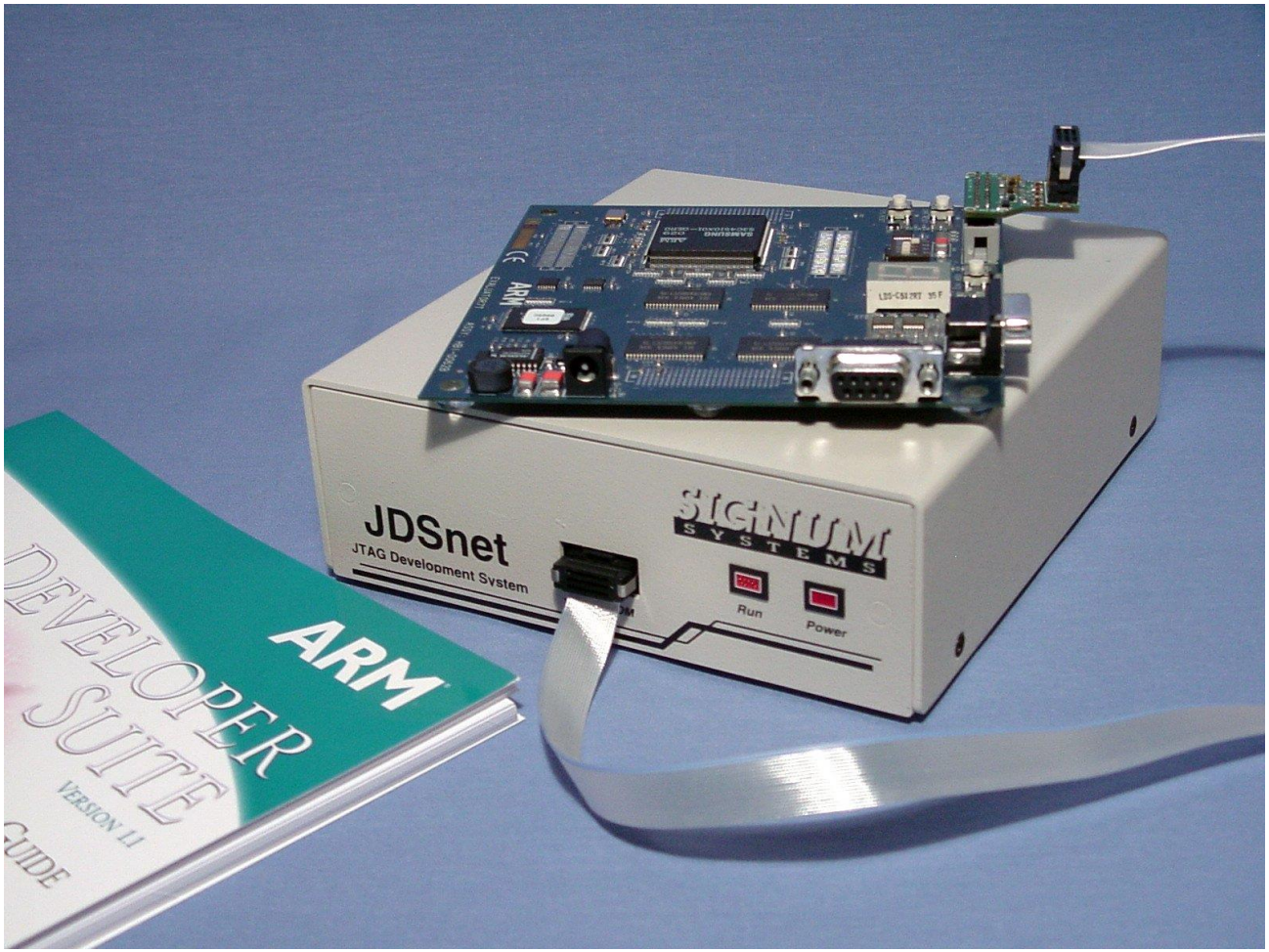
Debugging – ETB



Embedded Trace Buffer (ETB).

Embedded Trace Macrocell™
ETMv1.0 to ETMv3.4

JDSnet – ARM emulator



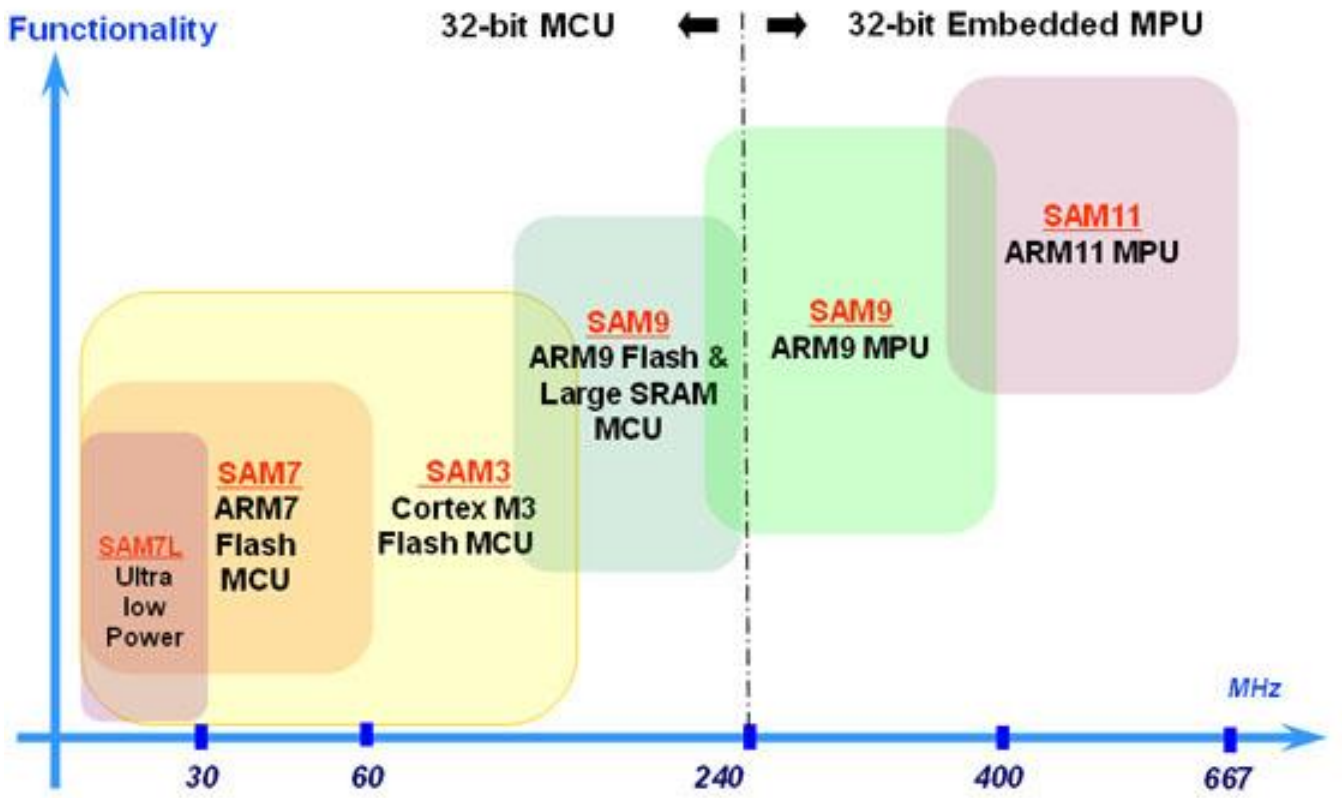
JDSnet – ARM emulator

- **JTAGjet-Trace -** a real-time, transparent in-circuit debugger with a real-time trace buffer.
- Supports all ARM7, ARM9, ARM11, Cortex, OMAP, OMAP2 and OMAP3 devices equipped with the Embedded Trace Macrocell (ETM) at speeds of up to 400 MHz.
- Available with trace depths of 256K, 1M, 2M and 4M frames and each frame is time-stamped with CPU cycle accuracy.

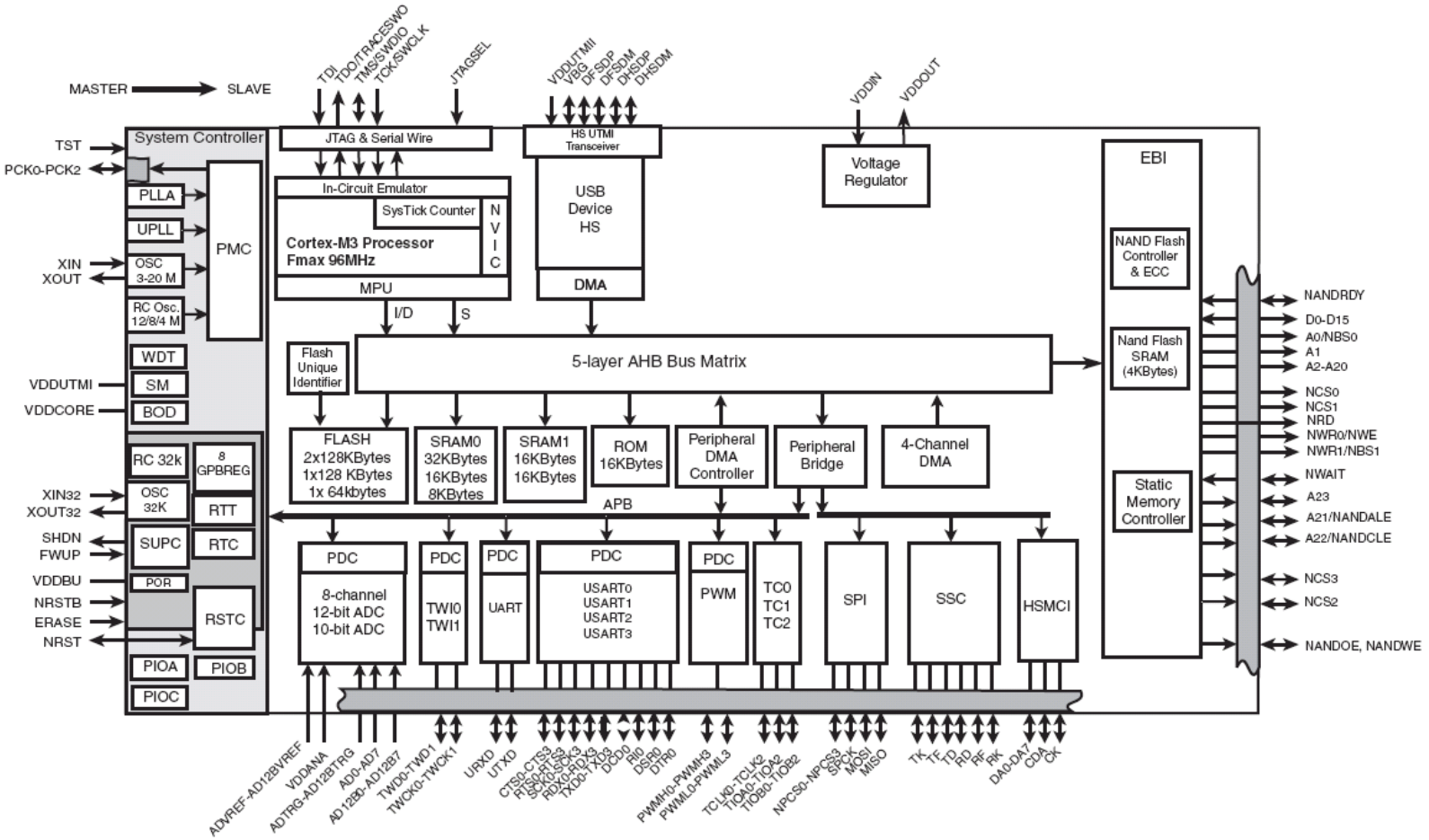
- Key Benefits:

  - Support for all ARM7, ARM9, ARM11, Cortex, OMAP, OMAP2 and OMAP3 cores equipped with ETM trace logic
  - Up to 400 MSamples/sec trace acquisition speed (up to 400 MHz CPU speeds)
  - Availability with 256K, 1M, 2M and 4M frame trace buffers
  - Support for 4, 8 and 16-bit wide trace busses
  - 56-bit time stamp with CPU cycle accuracy
  - Easy access to all ETM modes, triggers and filtering
  - Available with a standard Mictor-38 target connector or 20-pin Cortex connector
  - Quiet operation—no fans or exposed heatsinks
  - Plain JTAG adapter for boards without ETM

AT91SAM 32-bit ARM-based Microcontrollers



ARM Cortex-M3 – SAM3U4/2/1E Block Diagram



SAM3U4/2/1E

- **Memories**
  - **– From 64 to 256 Kbytes embedded Flash, 128-bit wide access, memory accelerator,**
  - **dual bank**
  - **– From 16 to 48 Kbytes embedded SRAM with dual banks**
  - **– 16 Kbytes ROM with embedded bootloader routines (UART, USB) and IAP routines**
  - **– Static Memory Controller (SMC): SRAM, NOR, NAND support. NAND Flash**
  - **controller with 4-kbyte RAM buffer and ECC**
- **System**
  - **– Embedded voltage regulator for single supply operation**
  - **– POR, BOD and Watchdog for safe reset**
  - **– Quartz or resonator oscillators: 3 to 20 MHz main and optional low power 32.768 kHz for RTC or device clock.**
  - **– High precision 8/12 MHz factory trimmed internal RC oscillator with 4 MHz Default Frequency for fast device startup**
  - **– Slow Clock Internal RC oscillator as permanent clock for device clock in low power mode**
  - **– One PLL for device clock and one dedicated PLL for USB 2.0 High Speed Device**
  - **– Up to 19 peripheral DMA (PDC) channels and 4-channel central DMA**

SAM3U4/2/1E

- **Low Power Modes**
  - – Sleep and Backup modes, down to 2.5 µA in Backup mode.
  - – Backup domain: VDDBU pin, RTC, 32 backup registers
  - – Ultra low power RTC: 0.6 µA
- **Peripherals**
  - – USB 2.0 Device: 480 Mbps, 4-kbyte FIFO, up to 7 bidirectional Endpoints,
  - dedicated DMA
  - – Up to 4 USARTs (ISO7816, IrDA®, Flow Control, SPI, Manchester support) and one
  - UART
  - – Up to 2 TWI (I2C compatible), 1 SPI, 1 SSC (I2S), 1 HSMCI (SDIO/SD/MMC)
  - – 3-Channel 16-bit Timer/Counter (TC) for capture, compare and PWM
  - – 4-channel 16-bit PWM (PWMC)
  - – 32-bit Real Time Timer (RTT) and RTC with calendar and alarm features
  - – 8-channel 12-bit 1Msps ADC with differential input mode and programmable gain stage, 8-channel 10-bit ADC
- **I/O**
  - – Up to 96 I/O lines with external interrupt capability (edge or level sensitivity), debouching, glitch filtering and on-die Series Resistor Termination
  - – Three 32-bit Parallel Input/Outputs (PIO)
- **Packages**
  - – 100-lead LQFP, 14 x 14 mm, pitch 0.5 mm
  - – 100-ball LFBGA, 9 x 9 mm, pitch 0.8 mm
  - – 144-lead LQFP, 20 x 20 mm, pitch 0.5 mm
  - – 144-ball LFBGA, 10 x 10 mm, pitch 0.8 mm

Performanțe

Figure 1. Relative performance for ARM7TDMI-S (ARM) and Cortex-M3 (Thumb-2)

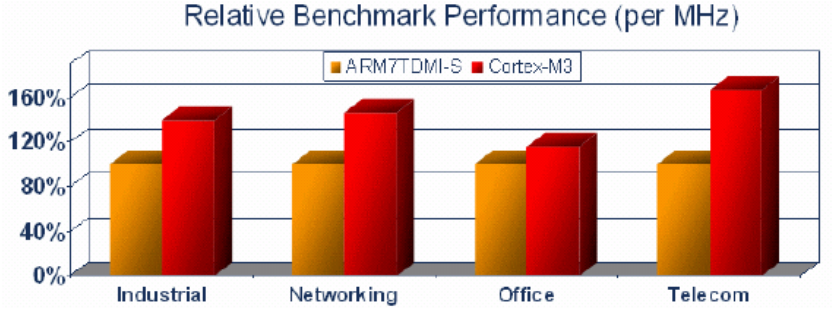
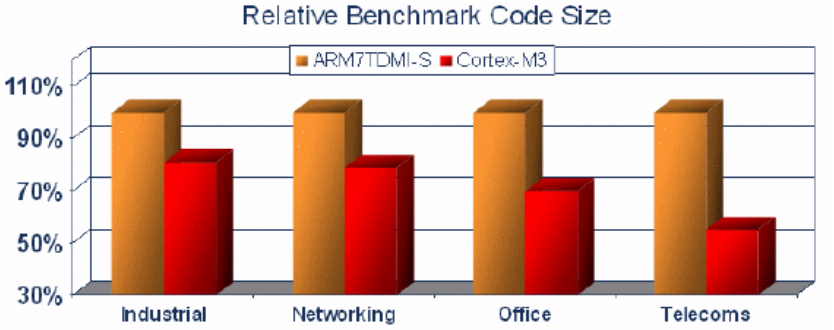


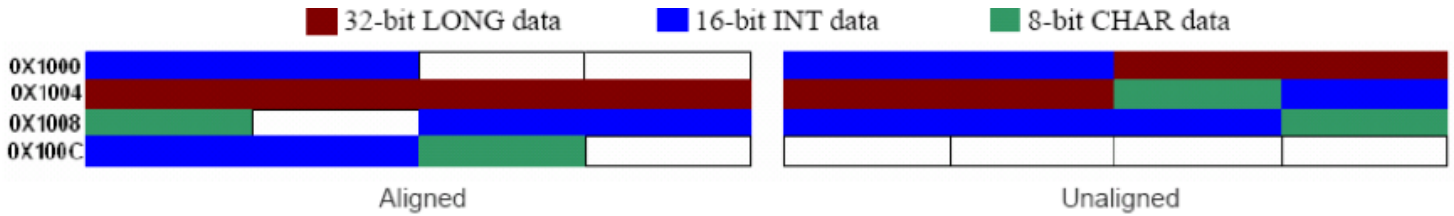
Figure 2. Relative code size for ARM7TDMI-S (ARM) and Cortex-M3 (Thumb-2)



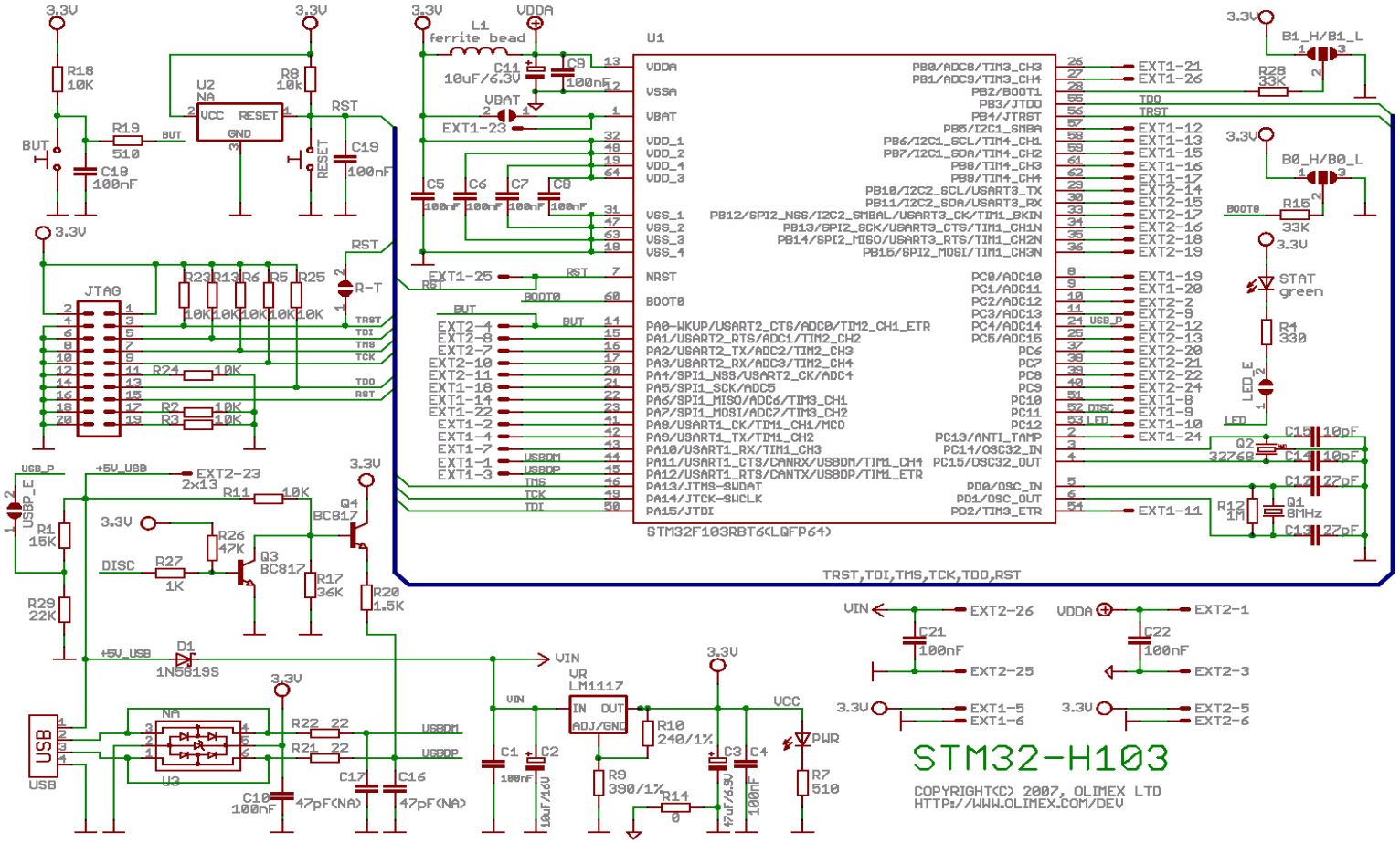
Performanțe

Features	ARM7TDMI-S	Cortex-M3
Architecture	ARMv4T (von Neumann)	ARMv7-M (Harvard)
ISA Support	Thumb / ARM	Thumb / Thumb-2
Pipeline	3-Stage	3-Stage + branch speculation
Interrupts	FIQ / IRQ	NMI + 1 to 240 Physical Interrupts
Interrupt Latency	24-42 Cycles	12 Cycles
Sleep Modes	None	Integrated
Memory Protection	None	8 region Memory Protection Unit
Dhrystone	0.95 DMIPS/MHz (ARM mode)	1.25 DMIPS/MHz
Power Consumption	0.28mW/MHz	0.19mW/MHz
Area	0.62mm2 (Core Only)	0.86mm2 (Core & Peripherals)*

Performanțe

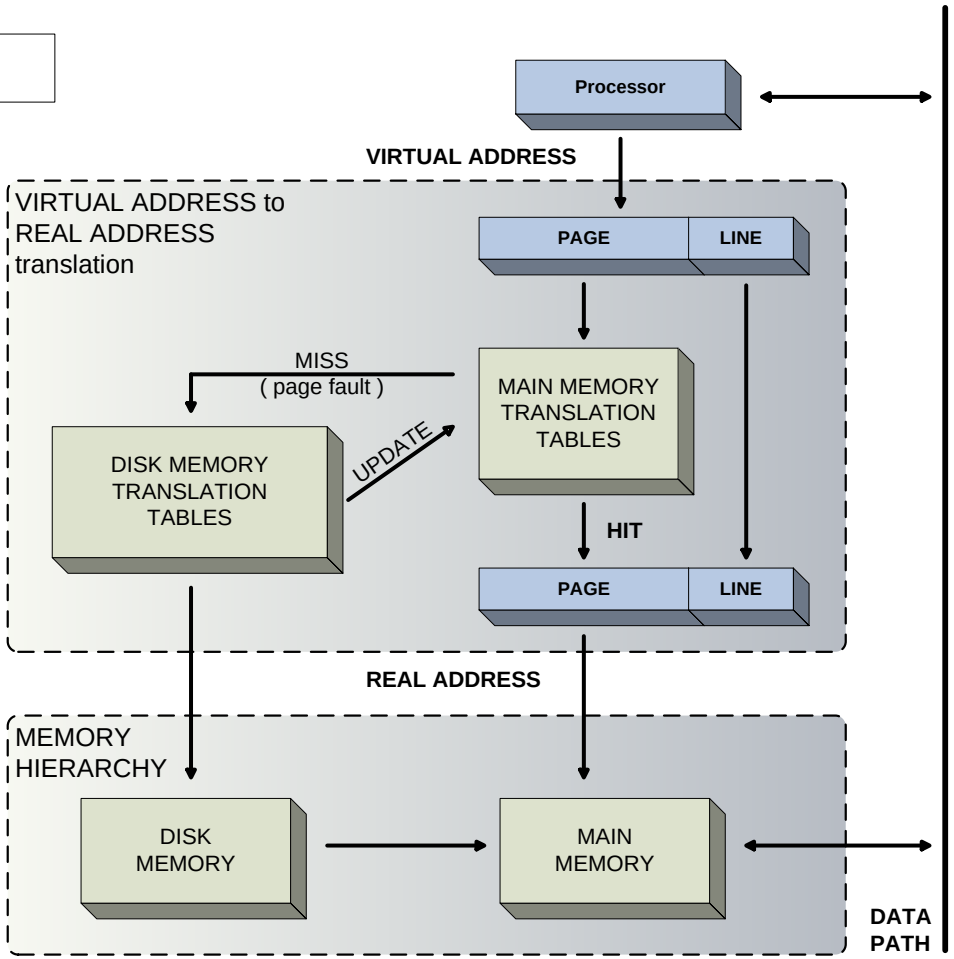


Sistem de dezvoltare cu ARM Cortex-M3



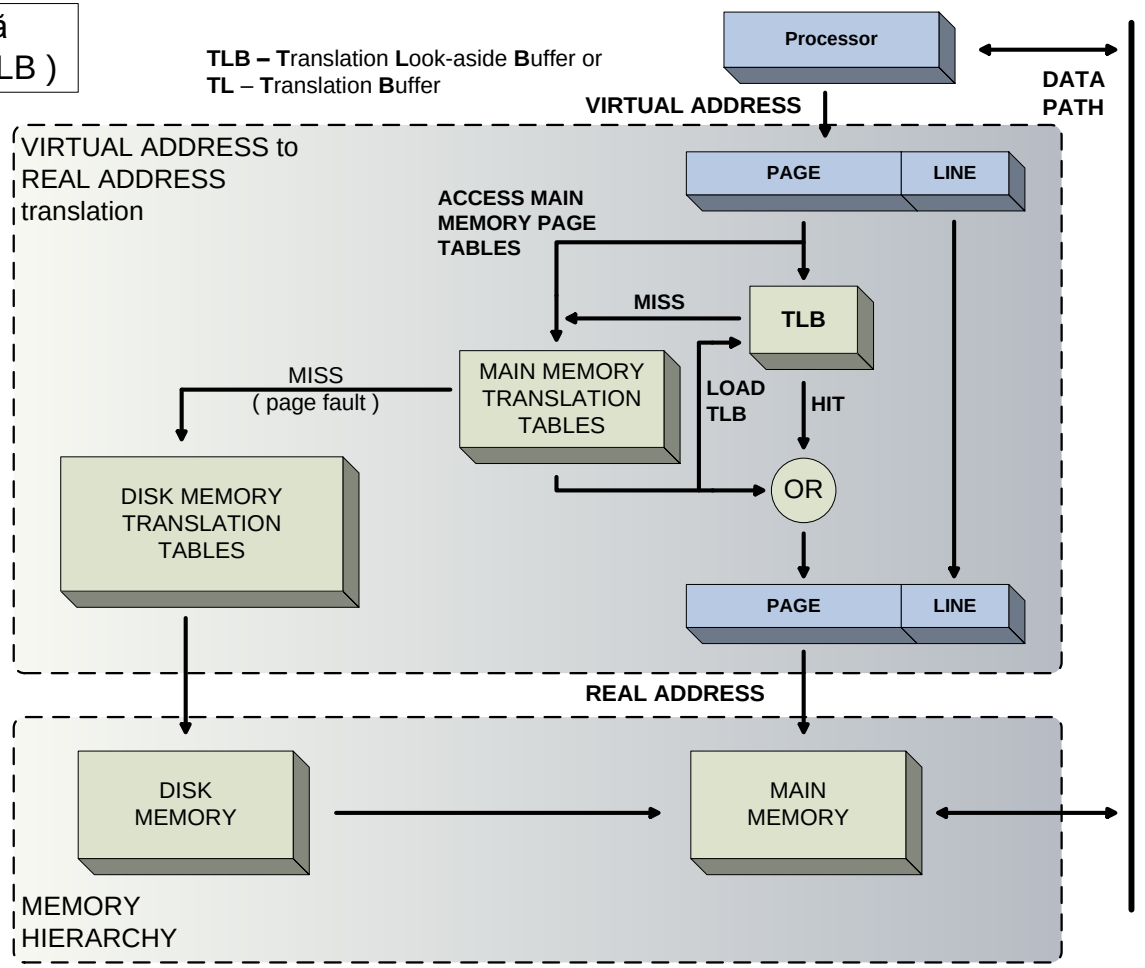
Structura şi organizarea calculatoarelor

Memoria virtuală



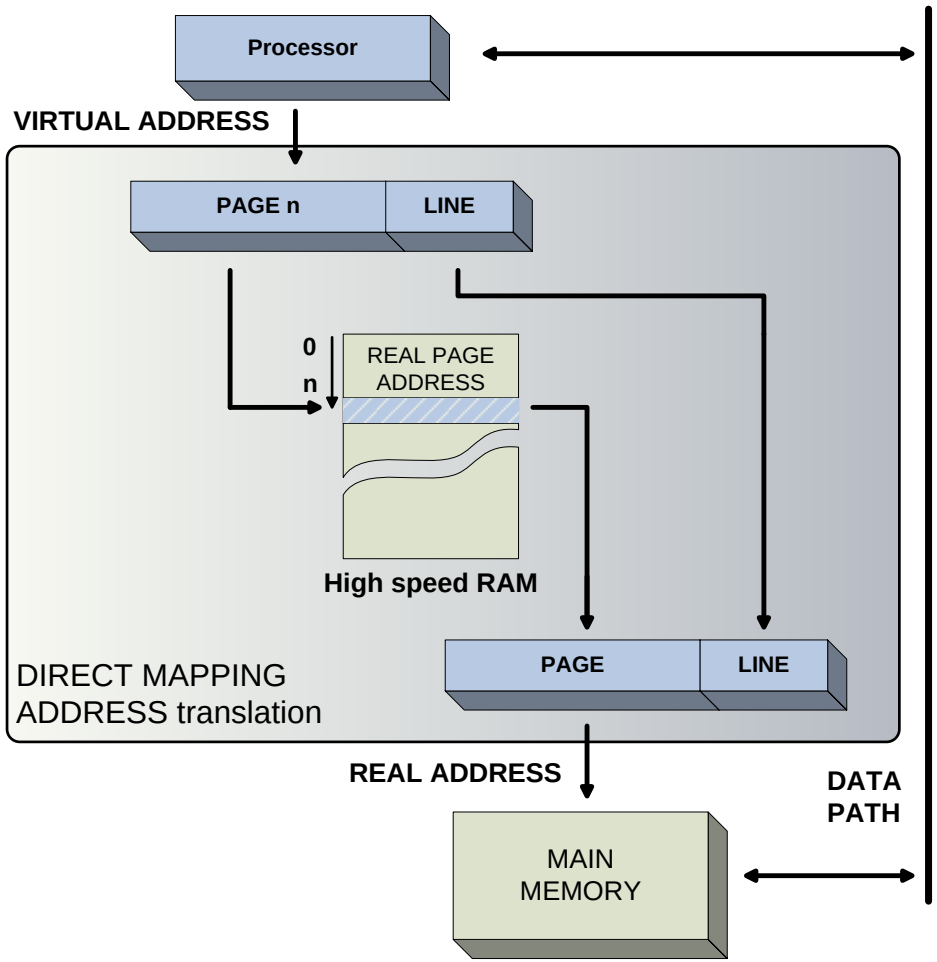
Memoria virtuală

Memoria virtuală
(translatare prin TLB)



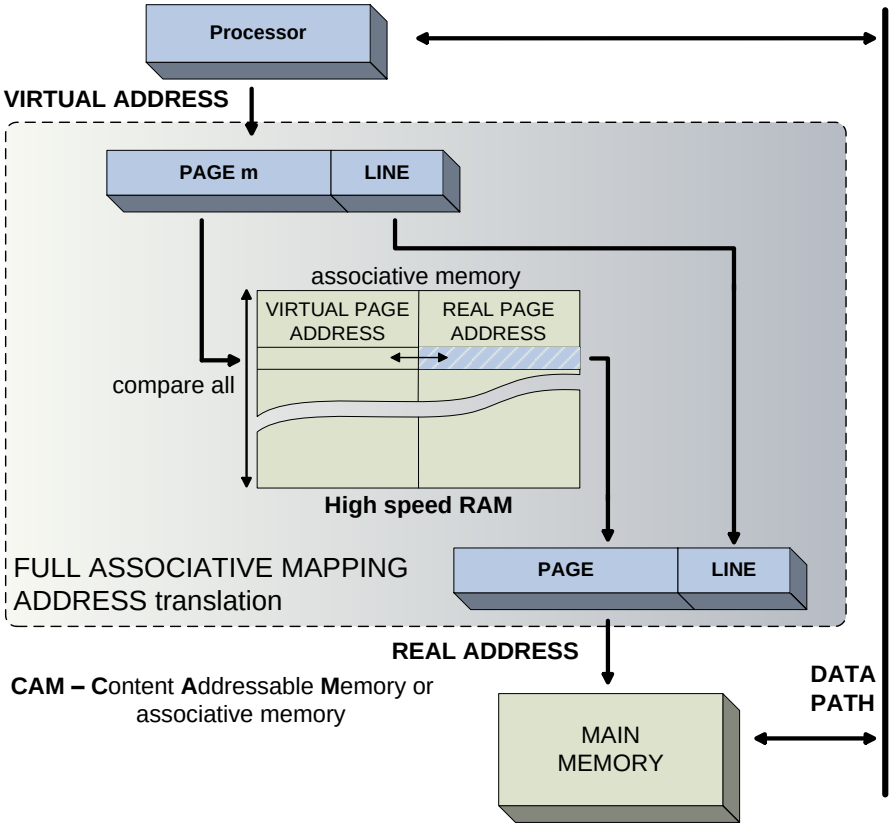
Memoria virtuală

Mapare directa



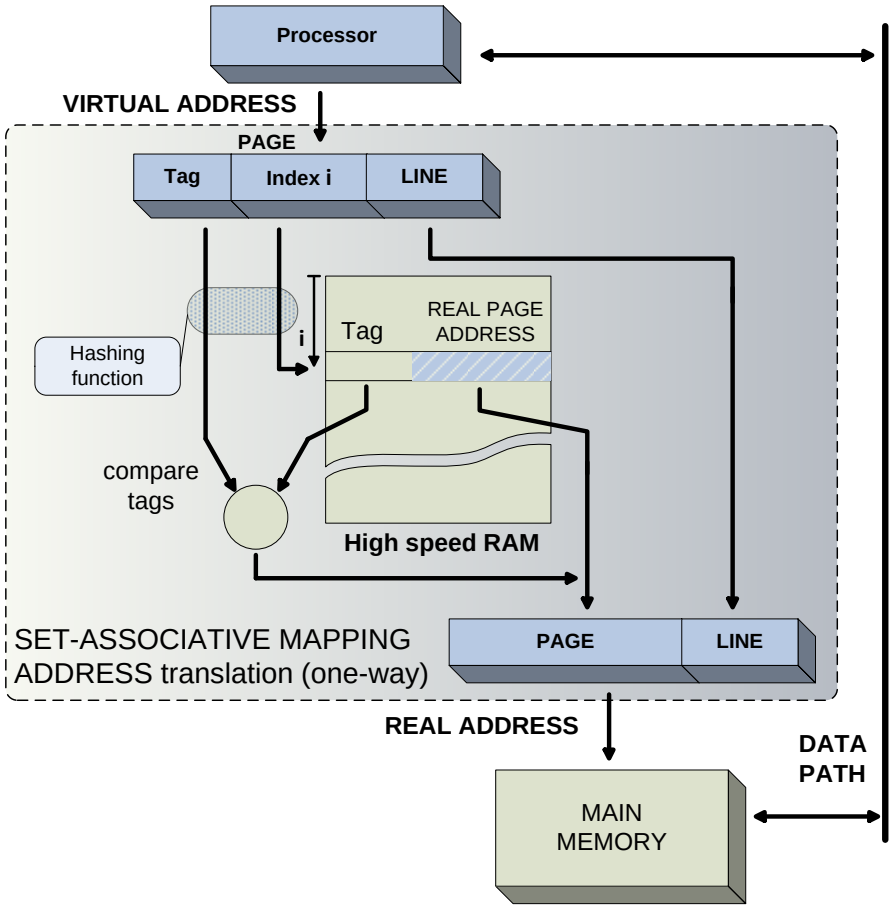
Memoria virtuală

Mapare asociativă



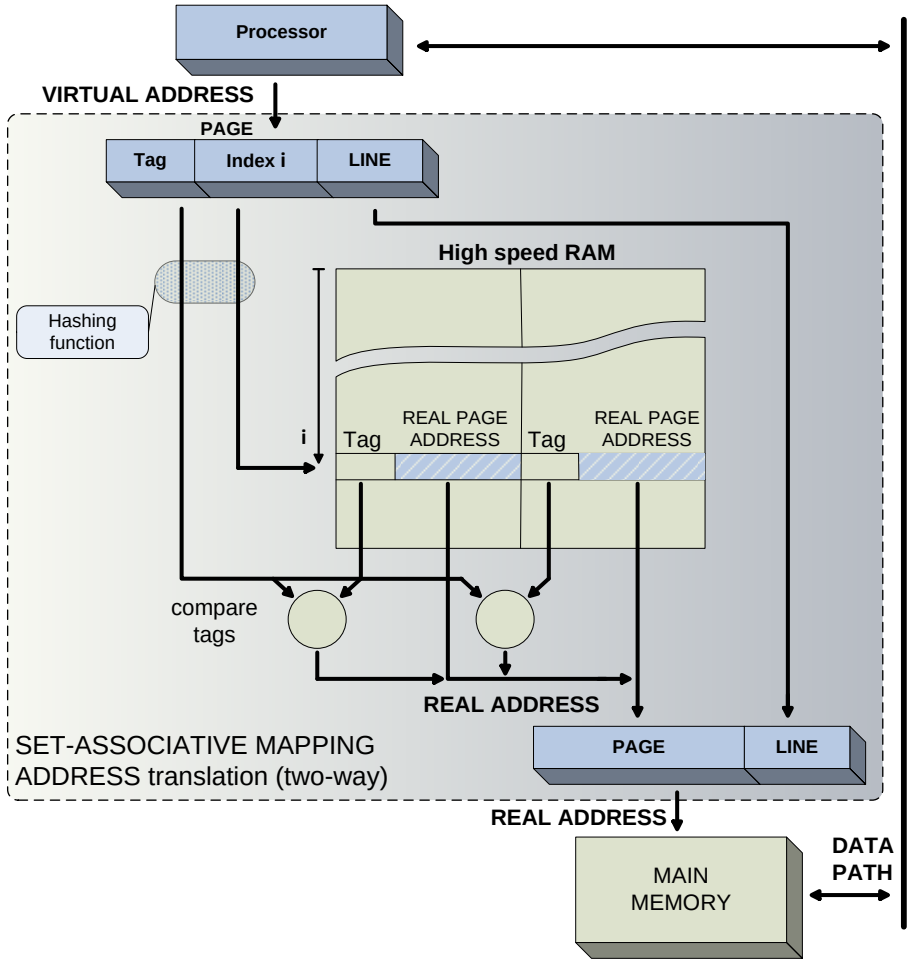
Memoria set-asociativă

Mapare set-asociativă
(one-way)

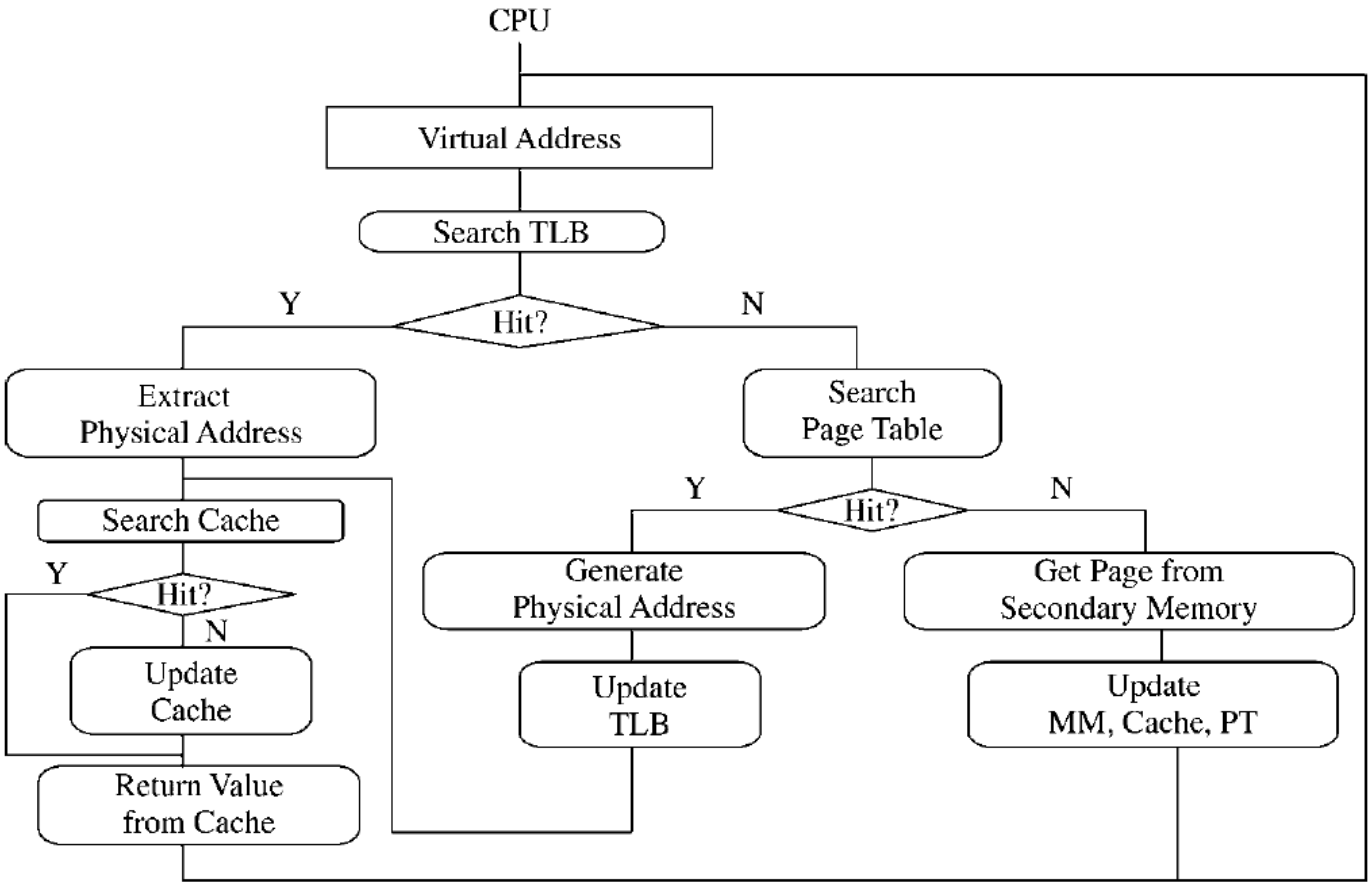


Memoria set-asociativă

Mapare set-asociativă
(two-way)



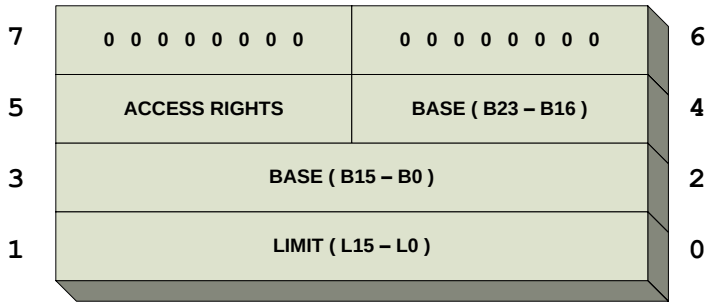
Accesul la memorie



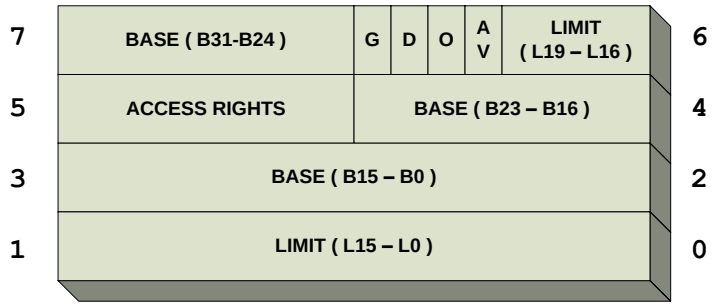
80x86 – Registre

Structura registrelor de identificare pentru 80X86, Pentium și Pentium Pro

80286 descriptor

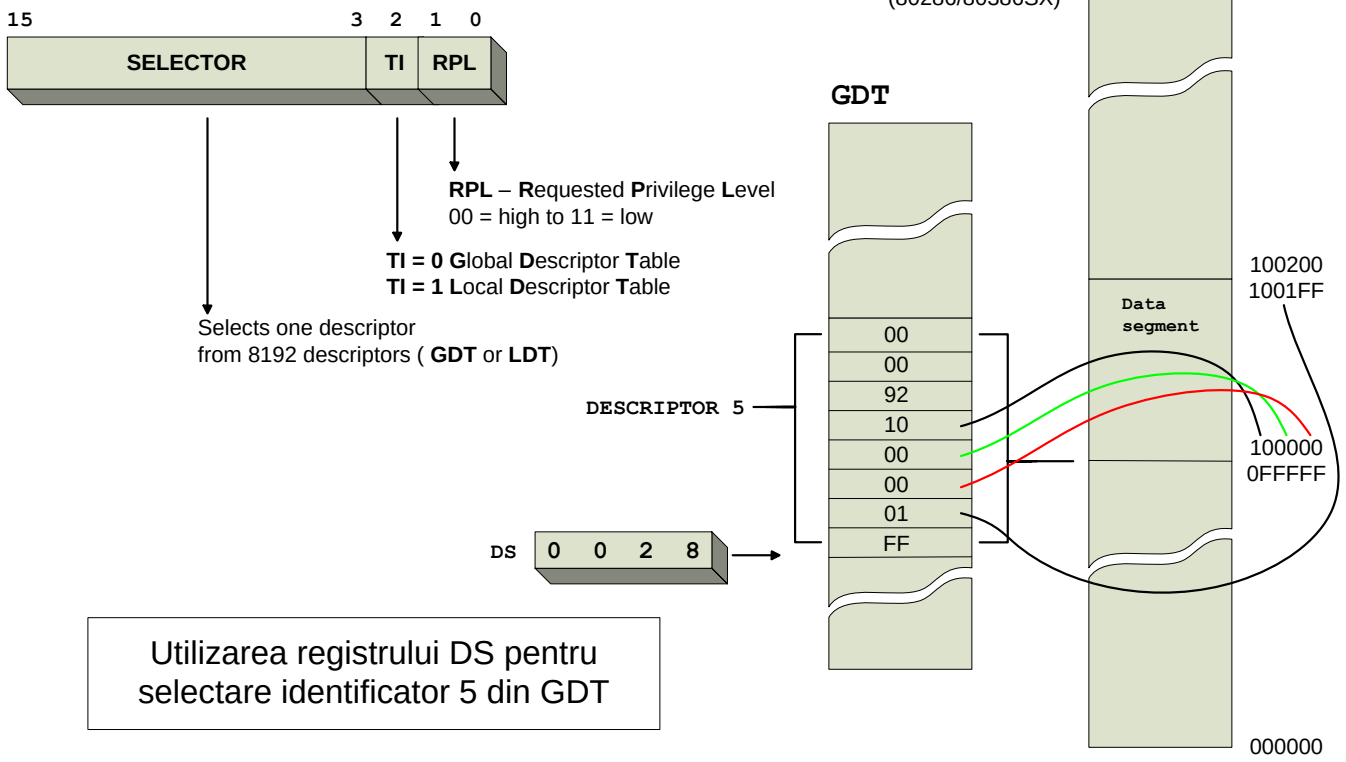


80386/80486/Pentium/Pentium Pro descriptor



80x86 – Registre

Structura registrului de segment (DS) în modul protejat

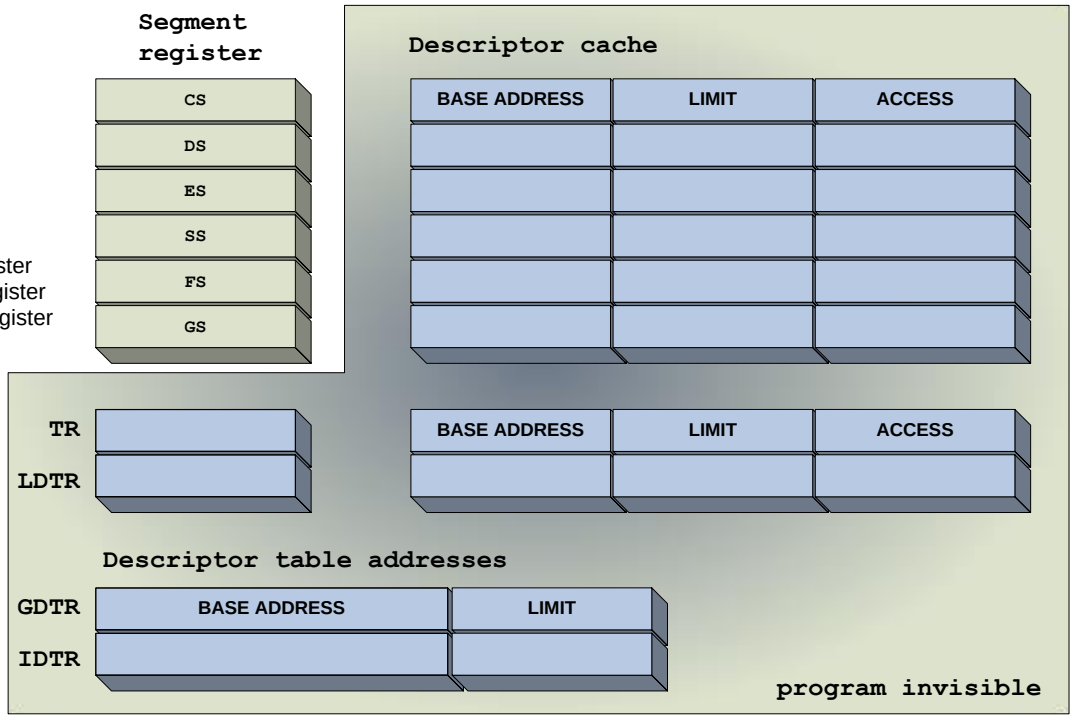


Utilizarea registrului DS pentru selectare identificator 5 din GDT

80x86 – Registre

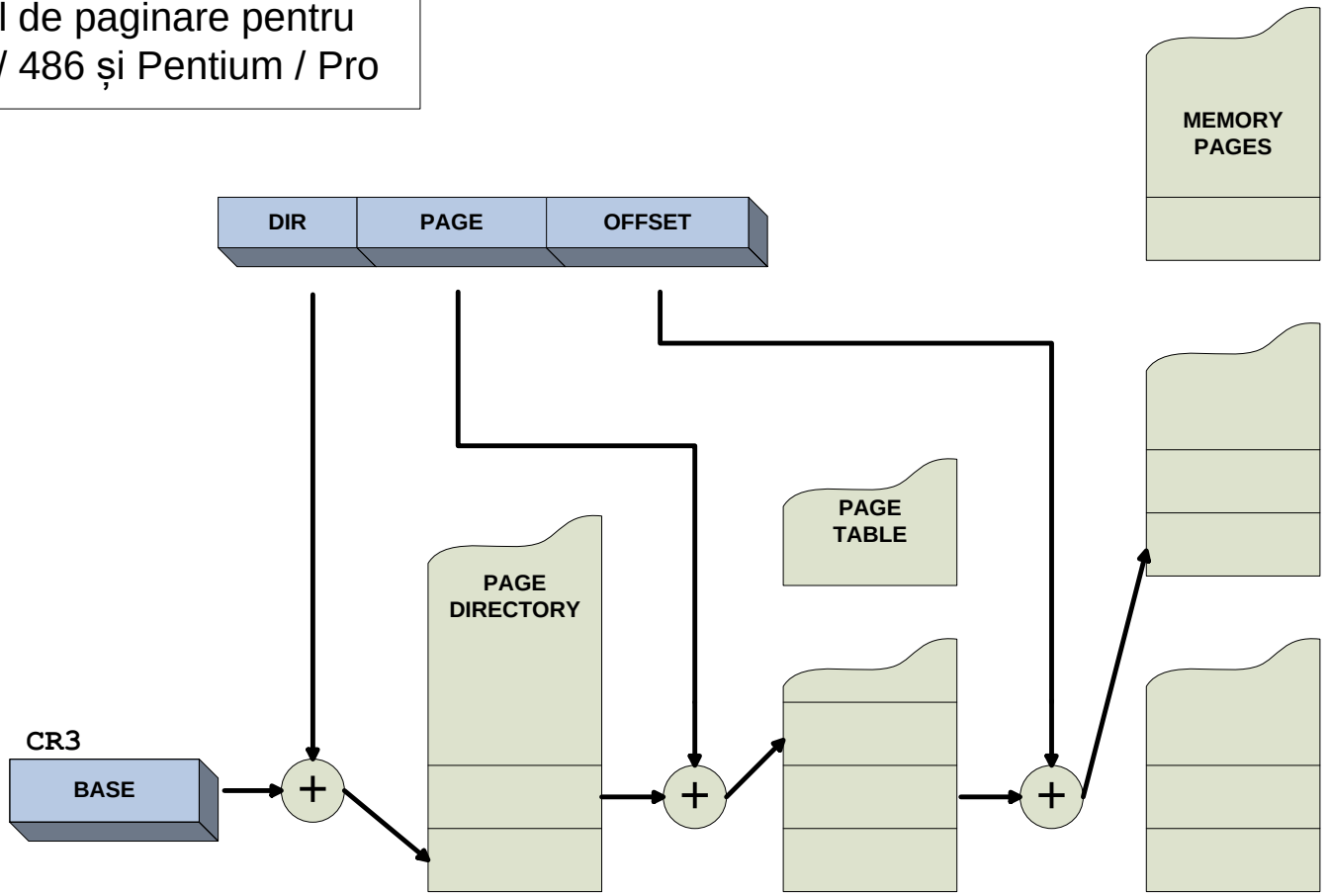
Harta registrelor “invizibili” în modul protejat pentru 80286 / 386 / 486 si Pentium / Pro

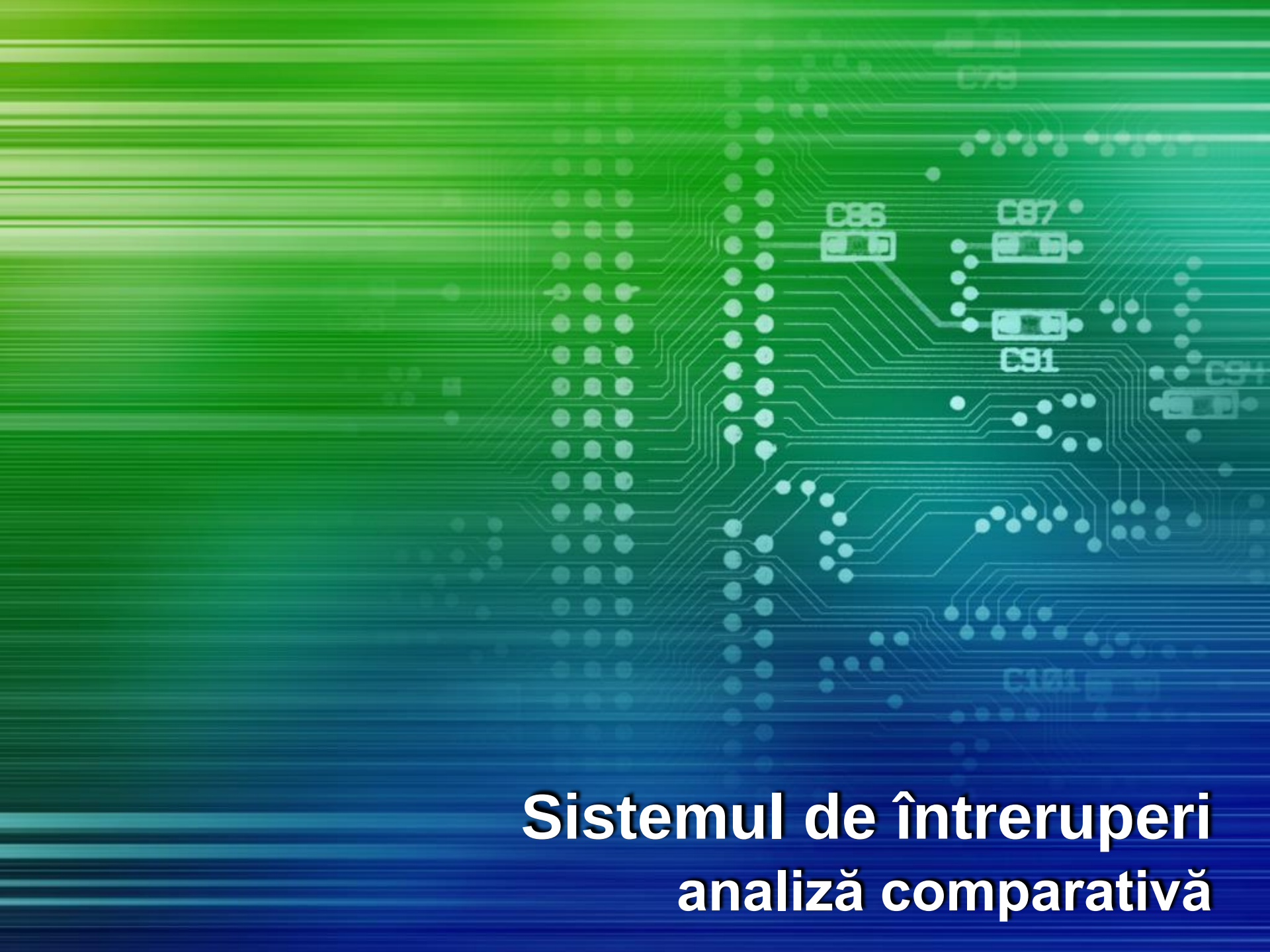
TR – Task **R**egister
LDTR – Local **D**escriptor **T**able **R**egister
GDTR – Global **D**escriptor **T**able **R**egister
IDTR – Interrupt **D**escriptor **T**able **R**egister



Paginarea

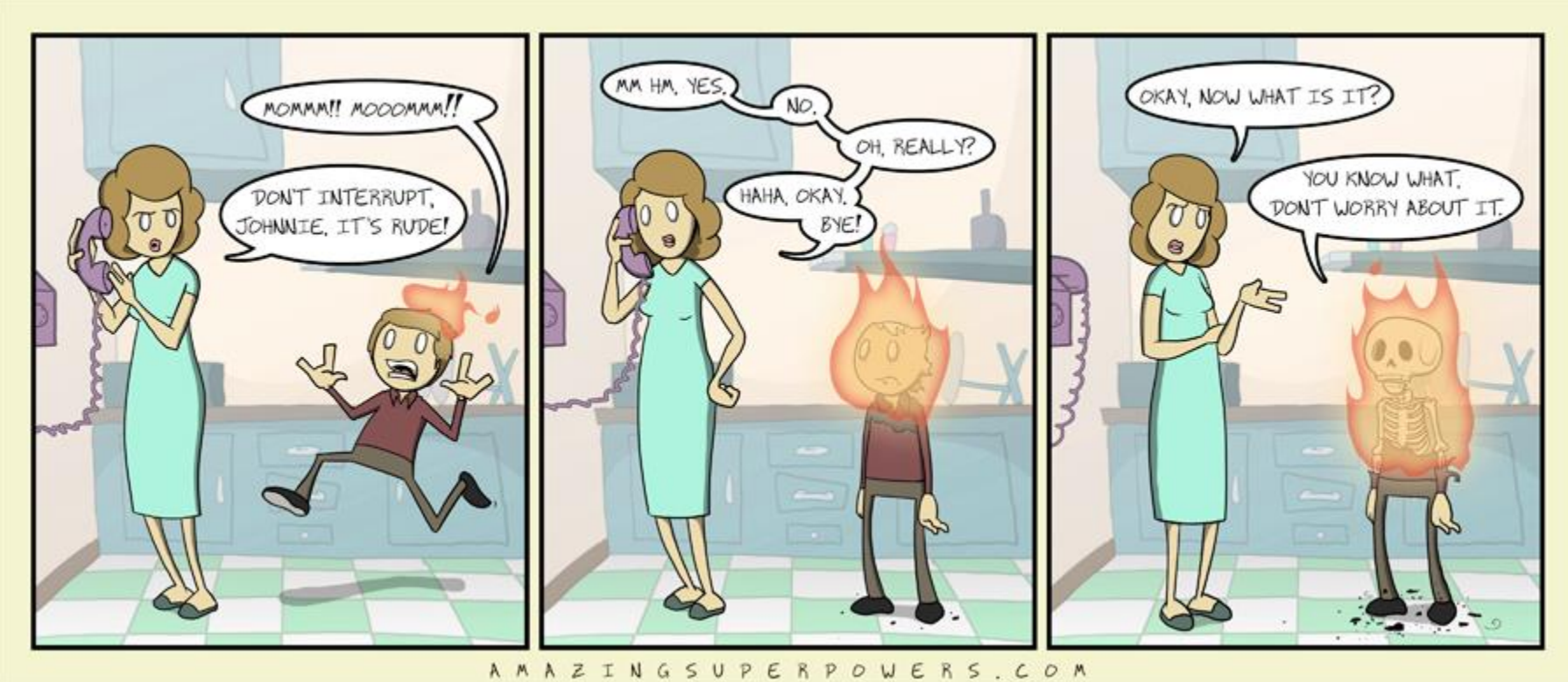
Modul de paginare pentru 80386 / 486 și Pentium / Pro





Sistemul de întreruperi
analiză comparativă

How *not* to use **interrupts**…



Sistemul de întreruperi

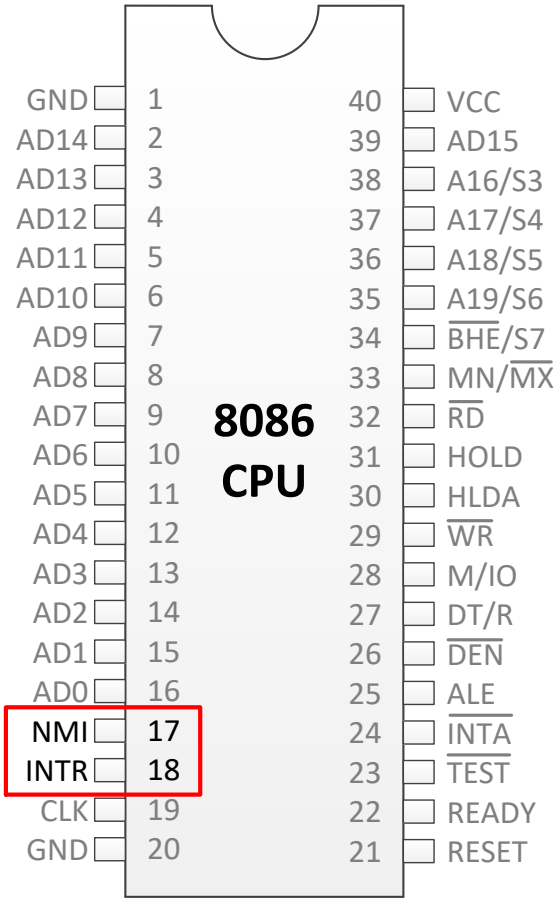
- Mecanism hardware & software
  - **detecția evenimentelor și reacția asociată**
- Evenimente
  - **interne/externe (în vorbire, trebuie specificat: relativ la CPU sau la MCU)**
  - **importanța ev. stabilește prioritatea de tratare a ev.**
  - **prioritatea de tratare a ev. este direct proporțională cu viteza de reacție la apariția ev.**
- CPU întrerupe execuția programului
  - **pentru a reacționa asupra apariției ev.**
    - execută **Rutina de Tratare a Întreruperii** (ISR – *Interrupt Service Routine*)

Intel 8086

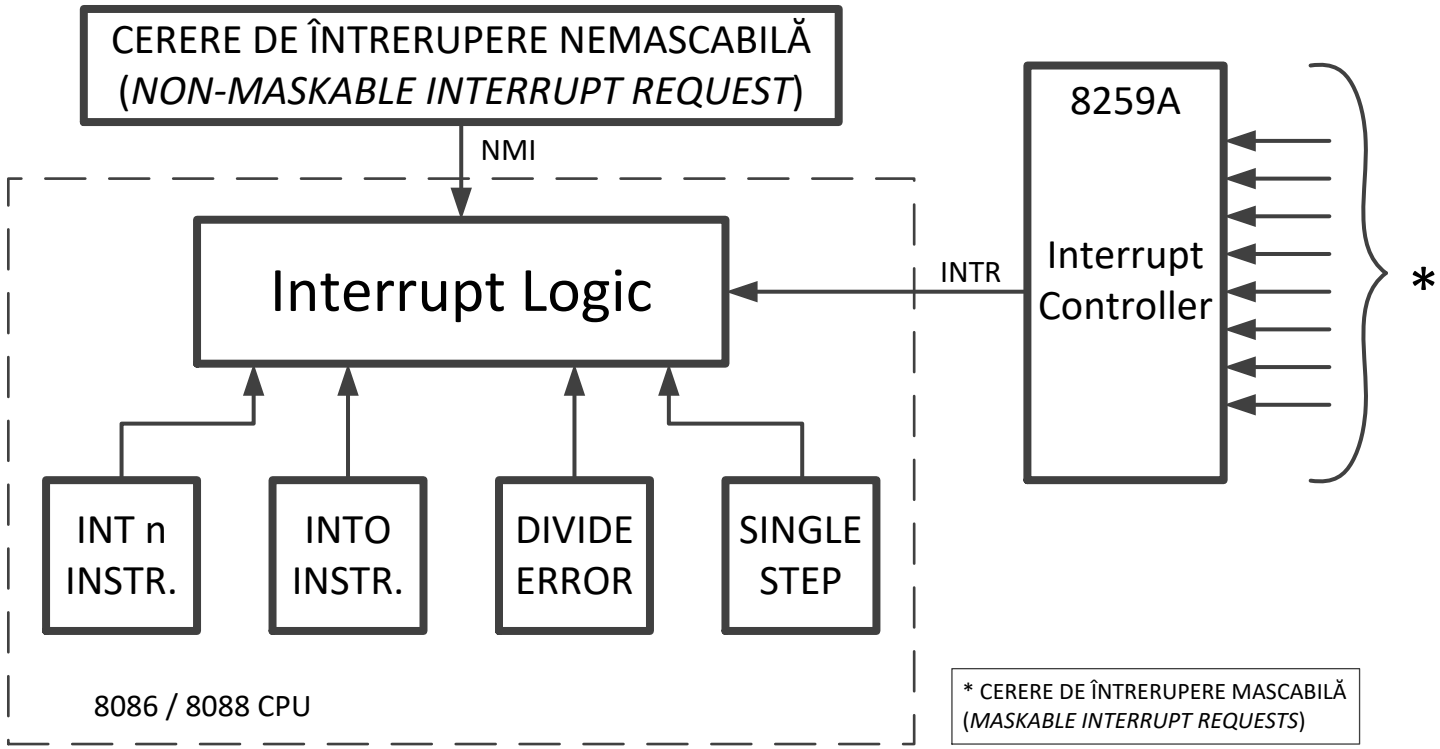
CPU

Intel 8086 CPU

- 1978, von Neumann, CISC, 99 instrucțiuni
- CPU dispune de 2 canale:
  - **NMI (Non-Maskable Interrupt)**
  - **INTR (Interrupt Request)**
- Expandare posibilă cu **PIC**:
  - **Programmable Interrupt Controller**
  - **până la 224 surse diferite**
- NMI – activ pe front
- INTR – activ pe palier



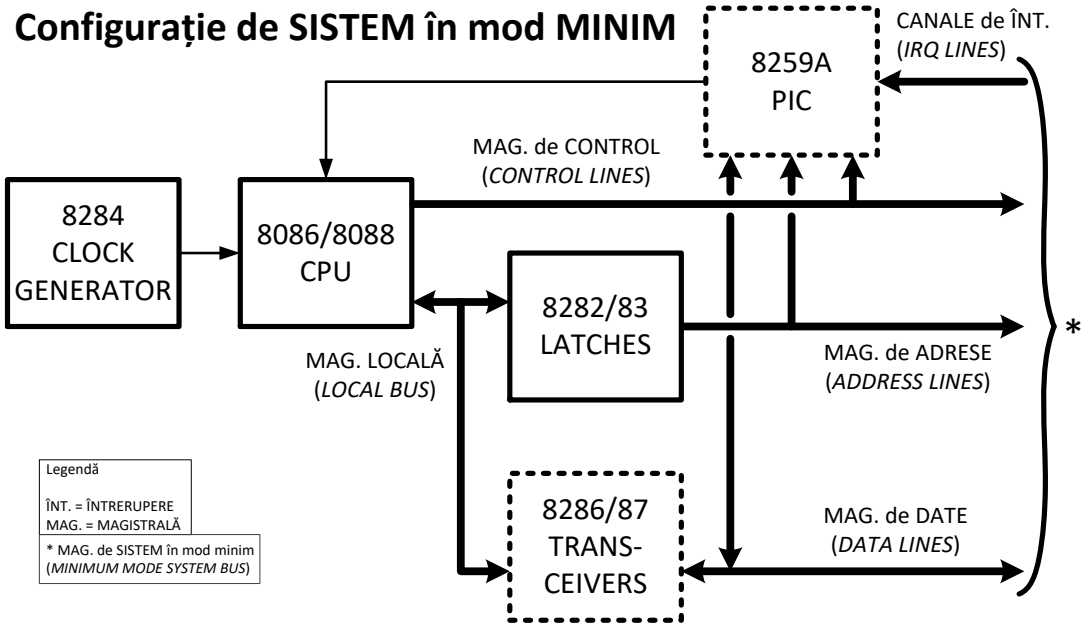
8086 CPU – Organizare Hardware



8086 CPU – Organizare Hardware (cont.)

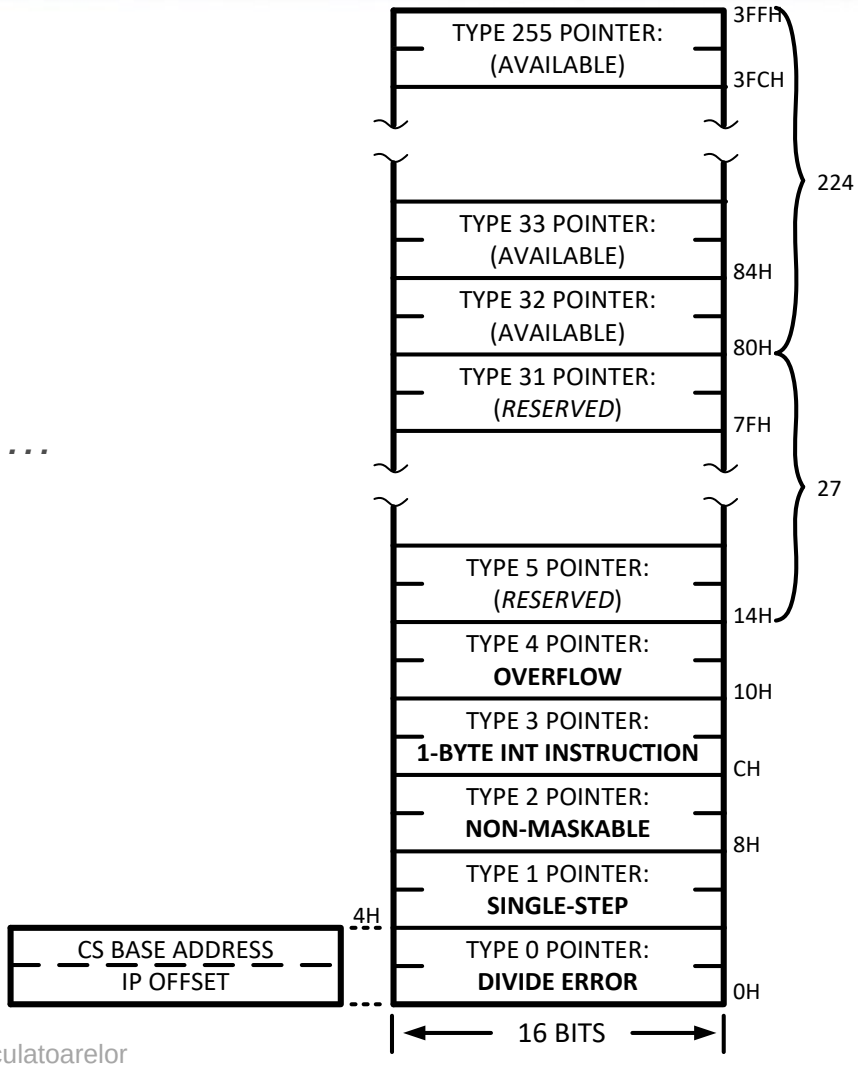
- Magistrală de sistem în mod minim (*Minimum mode system bus*)
  - magistrală controlată de un singur CPU

- Magistrală de sistem în mod maxim (*Multimaster system bus*)
  - comunicația poate fi inițiată de mai multe procesoare
  - magistrală *multimaster*
  - *bus controller* necesar



8086 CPU – Tabela vectorilor de întrerupere

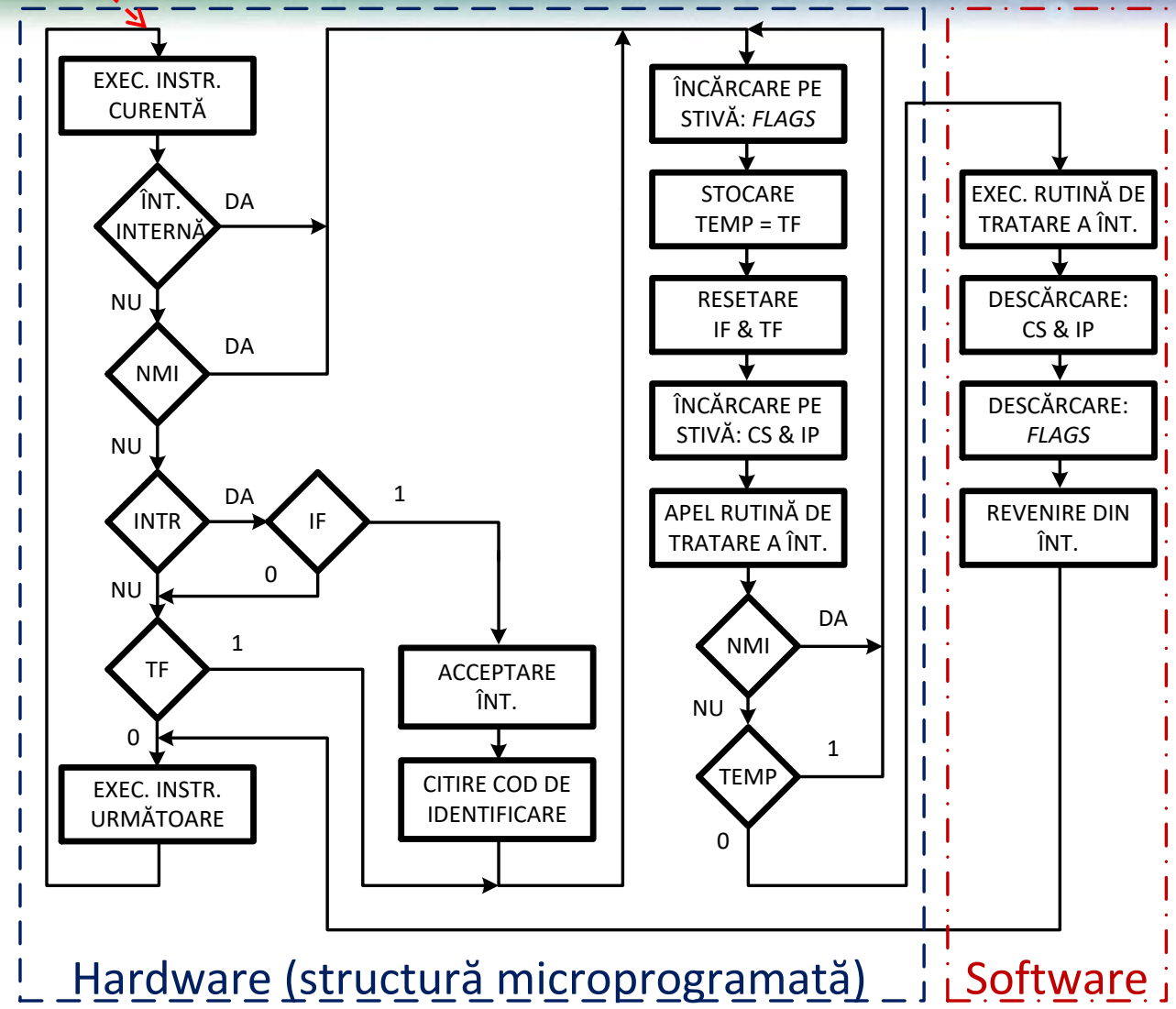
- Întreruperi vectorizate
- Asociere cod(*type*) ↔ eveniment
  - evenimente interne: *type 0, 1, 3, 4*
  - evenimente externe: *type 2, 32, 33 …*
  - interne/externe relativ la CPU
- CS + IP = adresa ISR
- Prioritățile evenimentelor
  - fixe la nivel de CPU
  - configurabile la nivel de PIC



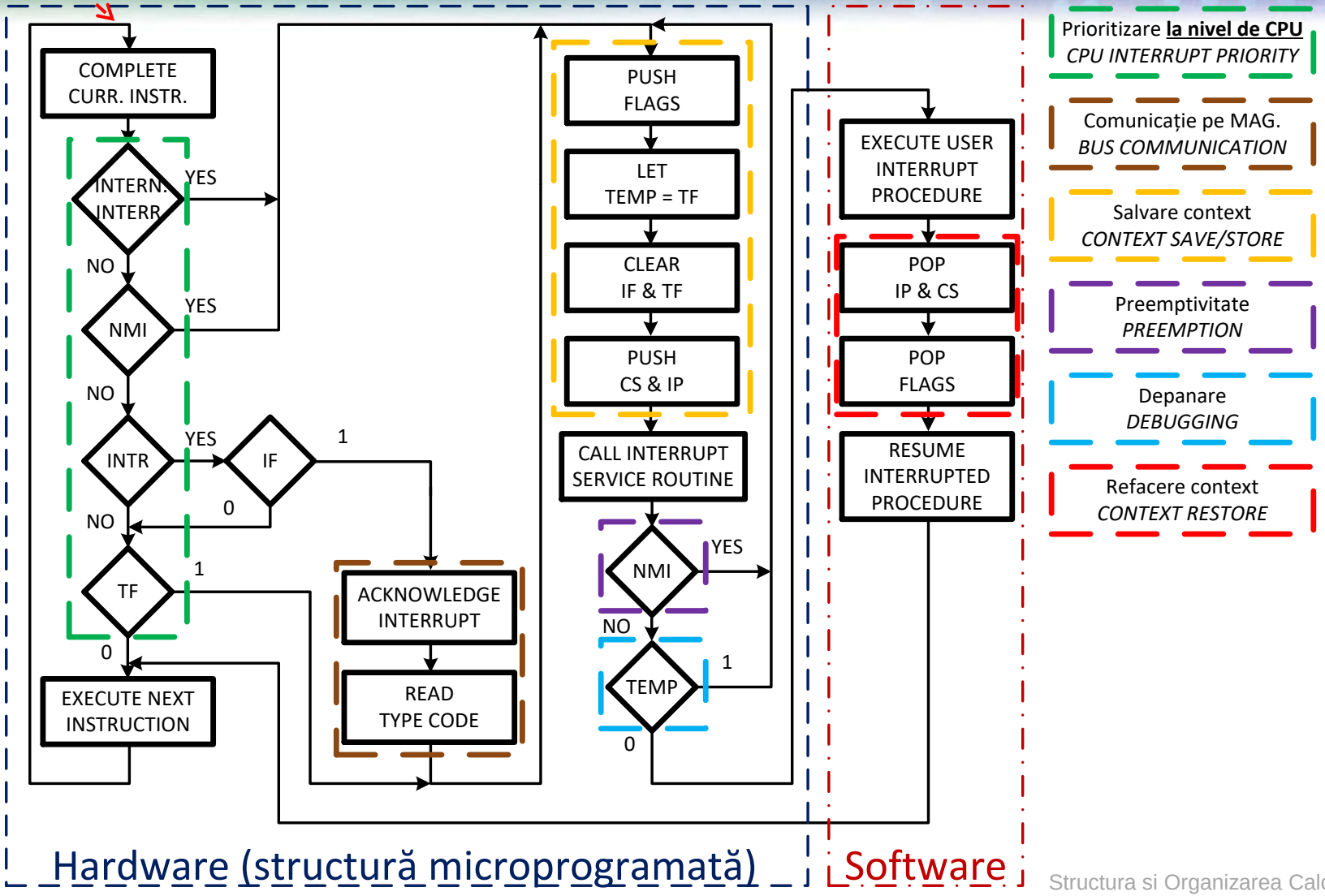
8086 – Prioritatea execuției întreruperilor

TIP	NUME	PRIORITATE
Internă	Divide Error, INT instruction (Software), INTO (Overflow)	Cea mai mare
Externă	NMI (Non Maskable Interrupt Request)	mare
Externă	INTR (Interrupt Request)	mică
Internă	Single-Step (Trap) – folosit ca instrument pentru depanare	Cea mai mică

8086 – Algoritmul de întrerupere



8086 – Algoritmul de întrerupere



Prioritizare **la nivel de CPU**
CPU INTERRUPT PRIORITY

Comunicație pe MAG.
BUS COMMUNICATION

Salvare context
CONTEXT SAVE/STORE

Preemptivitate
PREEMPTION

Depanare
DEBUGGING

Refacere context
CONTEXT RESTORE

Analiza timpului

- Timpul de procesare al întreruperii = timpul scurs de la detecția (≠ apariția) cererii până după salvarea **automată** (cablată) a contextului = T_{pro}
- T_{det} = timpul de la apariție pana la **det**ecție
- T_{sca} = timpul de **s**alvare **c**ontext **a**dițional (≠ automat)
- Latența întreruperii:
  - **timpul scurs de la apariția cererii până la prima instrucțiune de tratare (≠ salvare software de context)**

  = Tdet + Tpro + Tsca

8086 CPU – Timpul de procesare al întreruperii

Clasă de întrerupere	Timpul de procesare
INTR (Întrerupere Externă Mascabilă)	61 perioade
NMI (Întrerupere Externă Nemascabilă)	50 perioade
INT (cu vector)	51 perioade
INT Cod 3 (Software Interrupt – instrucțiunea INT)	52 perioade
INT Overflow	53 perioade
Single-Step (Debug)	50 perioade

8086 CPU – Latența întreruperii (nepreemptiv)

- Ipoteză de calcul:
  - **INTR, Tpro = 61; Tsca = 0**
  - **instrucțiunea curentă: IDIV durează 184 perioade**
- Cel mai scurt timp posibil:
  - **Cererea apare la sfârșitul execuției IDIV**
    - 61 + 1 = 62 perioade de CLK
- Cel mai lung timp posibil:
  - **Cererea apare la începutul execuției IDIV**
    - 61 + 184 = 245 perioade de CLK

8086 CPU – Referințe

- The 8086 Family User's Manual, Intel Corporation, October 1979
- The Intel Microprocessors 8th Edition, B.B. Brey, Pearson Education, 2009
- https://usermanual.wiki/Document/Intel8086FamilyUsersManual.2660397397/view
- https://userpages.umbc.edu/~squire/intel_book.pdf
- https://en.wikipedia.org/wiki/Intel_8086

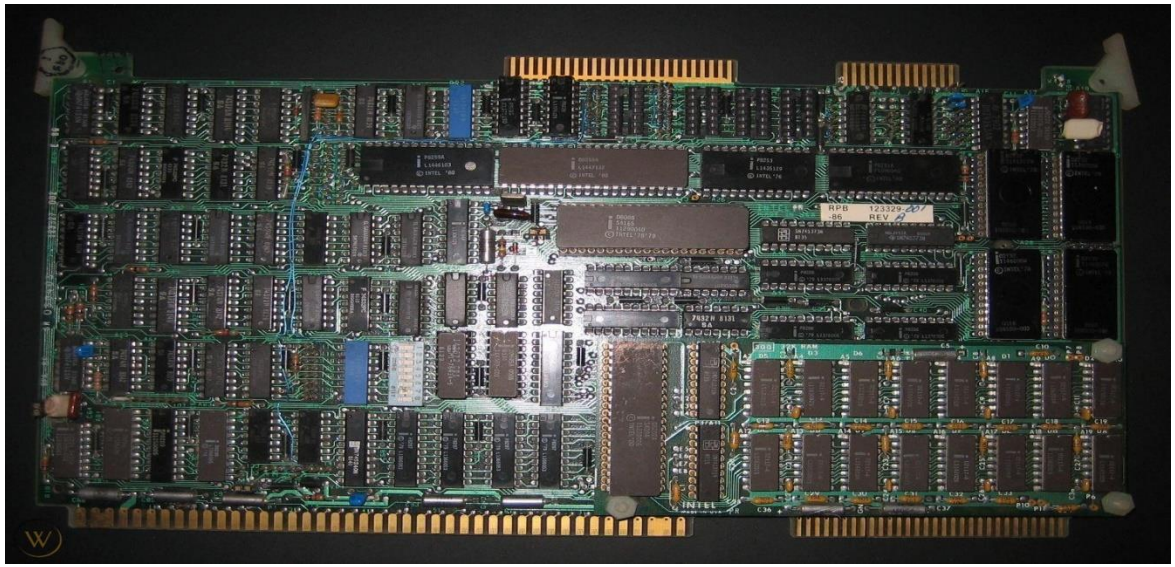
Implementări folosind 8086 CPU
MYCRO-1

- Printre primele SBC (*Single Board Computer*) : **MYCRO-1**
  - » (Raspberry Pi este un SBC)

https://en.wikipedia.org/wiki/MYCRO-1

- Cum arăta un single board computer:

https://www.worthpoint.com/worthopedia/intel-vintage-multibus-8086-cpu-1801654264



Implementări folosind 8086 CPU
Xerox NoteTaker

- Printre primele laptop-uri : **Xerox NoteTaker**

https://en.wikipedia.org/wiki/Xerox_NoteTaker

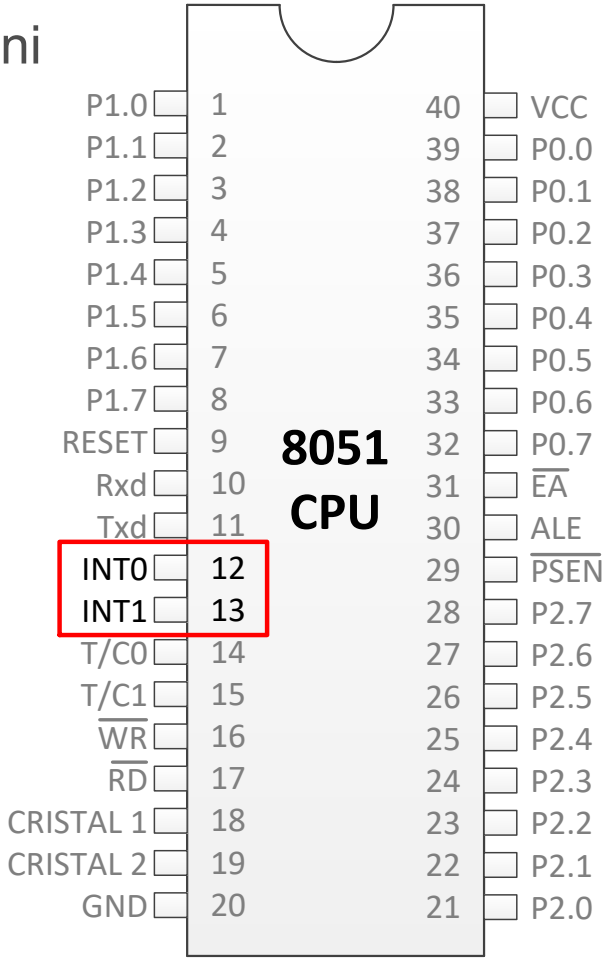
Xerox NoteTaker

The first laptop. But it couldnt really fit on your lap.

Intel 8051 Core

Intel 8051 Core – Generalități

- 1980, Harvard hibrid, CISC, 110 instrucțiuni
- Nucleu (*Core*) = CPU + periferice
- Nucleul dispozitivelor din familia MCS 51
- Aplicațiile țintă: sisteme încorporate
- Versiune inițială: tehnologie NMOS
- Versiunea CMOS: consum mai redus
  - **Aplicații cu alimentare de la baterie**



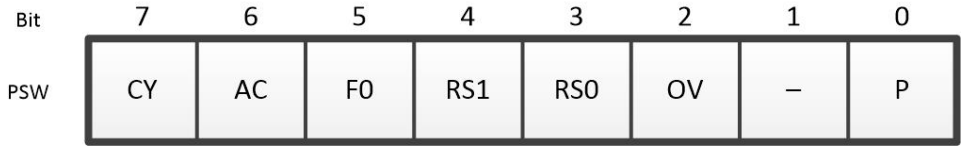
8051 Core – Întreruperi

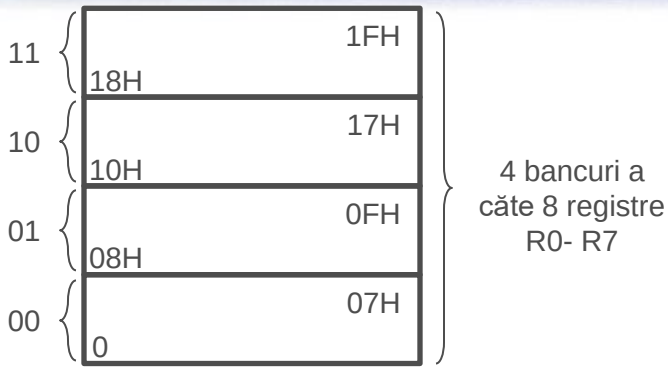
- 5 surse de întrerupere:
  - **2 externe (INT0 și INT1)**
  - **2 de la modulele Timer (TF0 și TF1)**
  - **1 de la modulul de comunicație UART (RI sau TI)**
- Activare/dezactivare individuală
- Prioritate configurabilă la nivel de nucleu
  - **prioritate pe 2 niveluri**
    - prin software, se poate adăuga al treilea nivel

8051 Core – Întreruperi

- INT0 si INT1
  - **întreruperi externe**
  - **active pe palier sau front**
  - **mascabile**
- Nu există întreruperi nemascabile
- Salvarea contextului pe stivă
  - **HW - microprogramat : doar registrul PC**
  - **SW - PUSH: PSW & registre de uz general – daca e necesar**
    - Posibilitate de reducere a timpului de salvare context
- Salvarea contextului din registre
  - **banc de registre dedicat tratării întreruperii**
  - **selecția bancului: în registrul PSW**

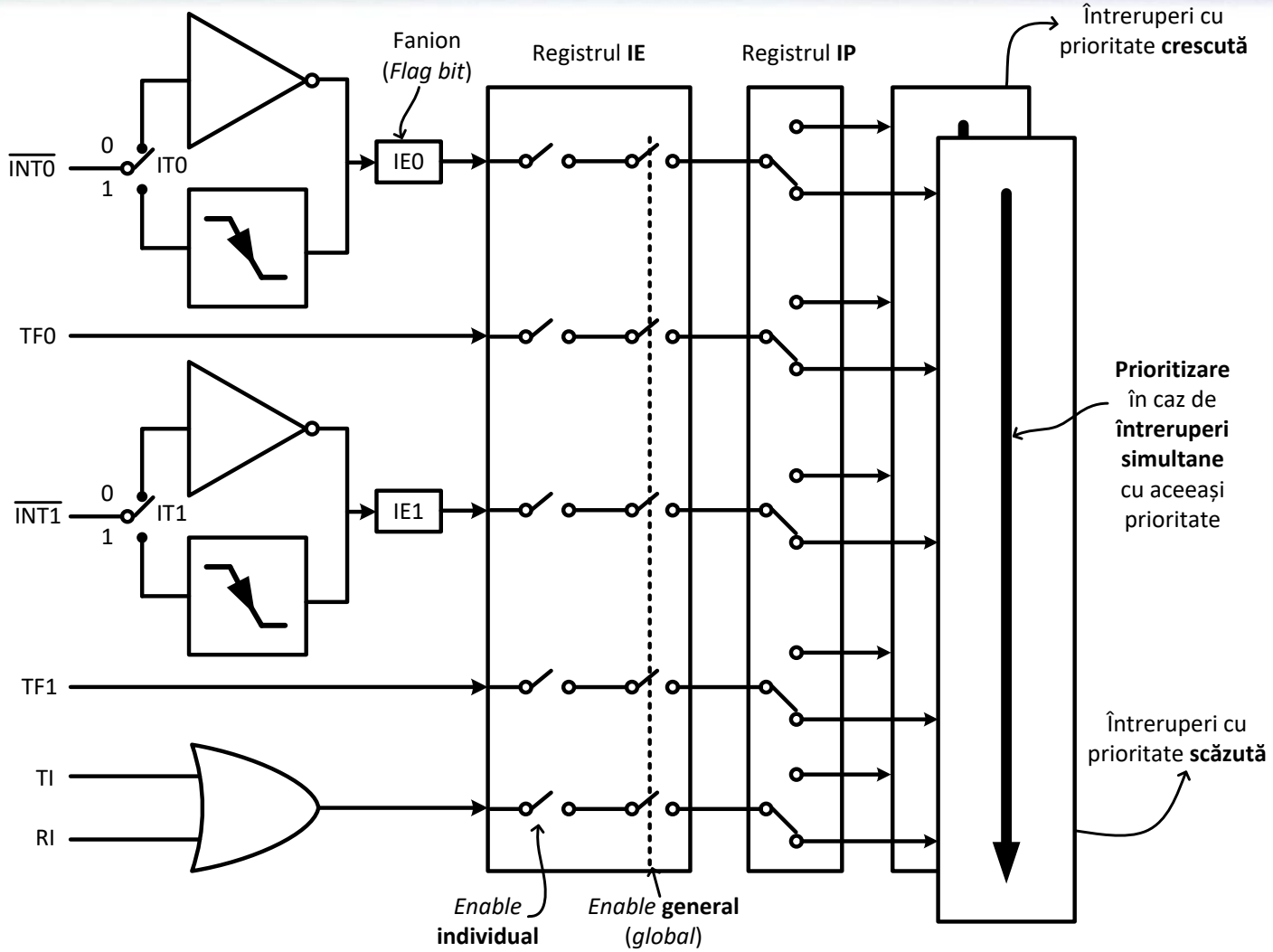
8051 Core – Întreruperi





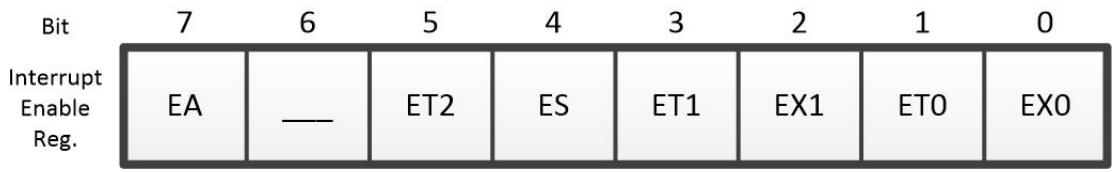
- CY, fanion de transport (*Carry Flag*)
- AC, fanion auxiliar de transport (*Auxiliary Carry Flag*)
- F0, semnificație stabilită de utilizator (*Flag 0)*
- **RS1, RS0, biți de selecție a bancului de registre (*Register Bank Selector*)**
- OV, depășire (O*verflow*)
- –, semnificație stabilită de utilizator
- P, fanion de paritate (*Parity Flag*)

8051 Core – Mecanismul de întreruperi



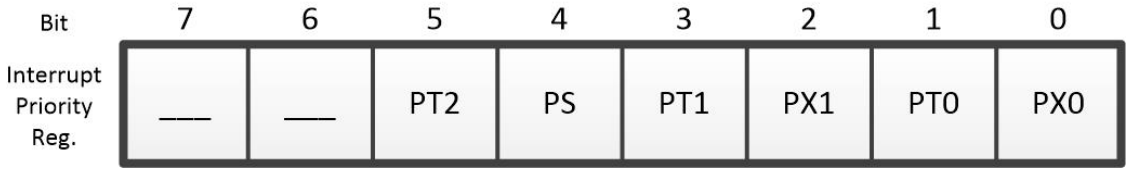
8051 Core – Activarea întreruperilor

SIMBOL	Funcția bitului din registrul IE (Interrupt Enable)	
EA	Activeaza/Dezactivează tratarea tuturor întreruperilor. EA = 0 – nicio întrerupere nu va fi tratată. EA = 1 – fiecare întrerupere va fi tratată dacă bitul aferent întreruperii este activ.	
___	Rezervat implementărilor viitoare. A nu se scrie valoarea 1.	
ET2	Cronometru 2 (Timer 2) (doar 8052)	activare/dezactivare (1/0)
ES	Port Serial – Comunicație serială (Serial Port)	activare/dezactivare (1/0)
ET1	Cronometru 1 (Timer 1)	activare/dezactivare (1/0)
EX1	Întrerupere externă 1 (External Interrupt 1)	activare/dezactivare (1/0)
ET0	Cronometru 0 (Timer 0)	activare/dezactivare (1/0)
EX0	Întrerupere externă 0 (External Interrupt 0)	activare/dezactivare (1/0)



8051 Core – Prioritatea întreruperilor

SIMBOL	Funcția bitului din registrul IP (Interrupt Priority)	
___	Rezervat implementărilor viitoare. A nu se scrie valoarea 1.	
___	Rezervat implementărilor viitoare. A nu se scrie valoarea 1.	
PT2	Cronometru 2 (Timer 2) (doar 8052)	prioritate mare/mică (1/0)
PS	Port Serial – Comunicație serială (Serial Port)	prioritate mare/mică (1/0)
PT1	Cronometru 1 (Timer 1)	prioritate mare/mică (1/0)
PX1	Întrerupere externă 1 (External Interrupt 1)	prioritate mare/mică (1/0)
PT0	Cronometru 0 (Timer 0)	prioritate mare/mică (1/0)
PX0	Întrerupere externă 0 (External Interrupt 0)	prioritate mare/mică (1/0)

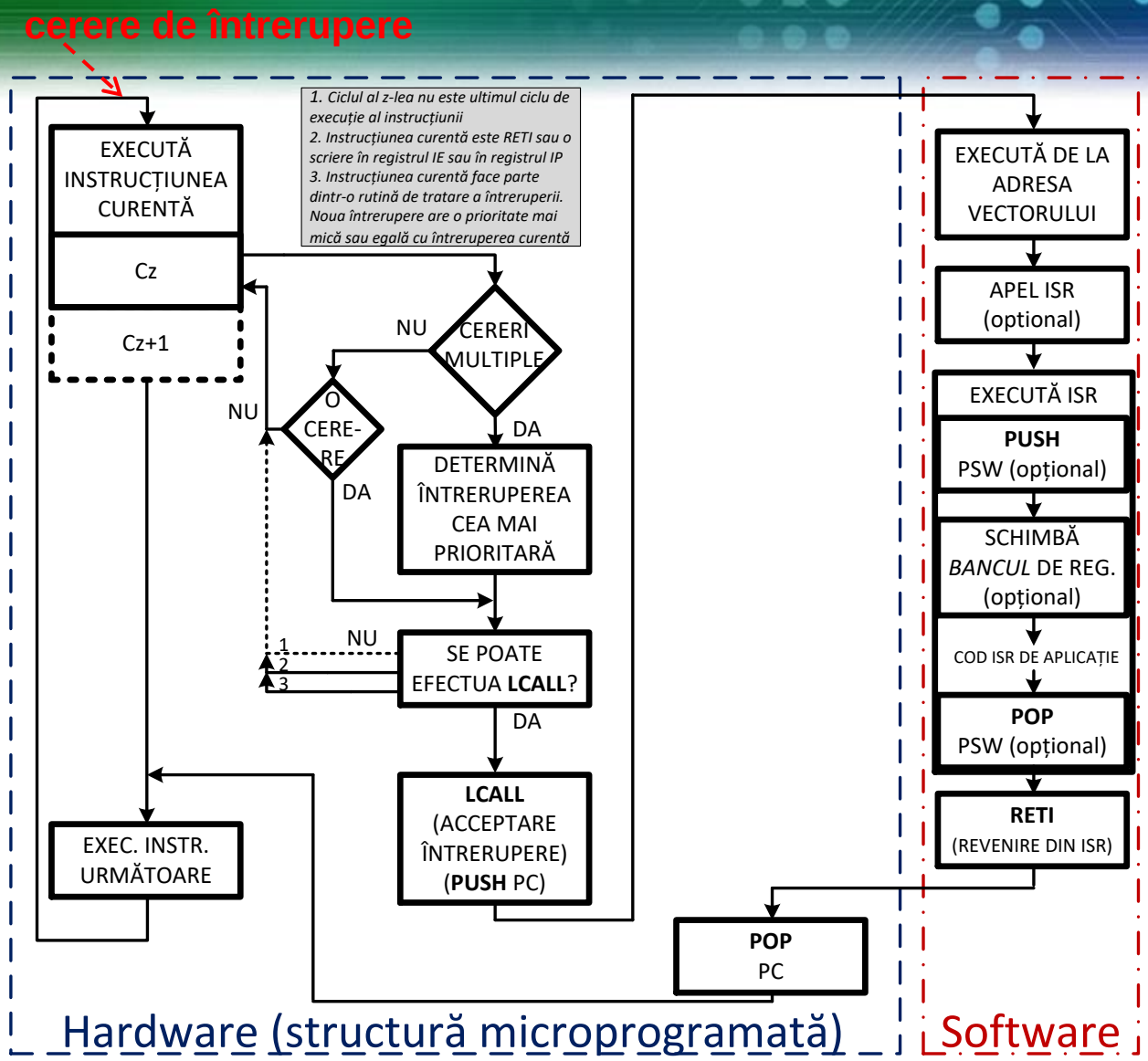


8051 Core – Prioritatea întreruperilor

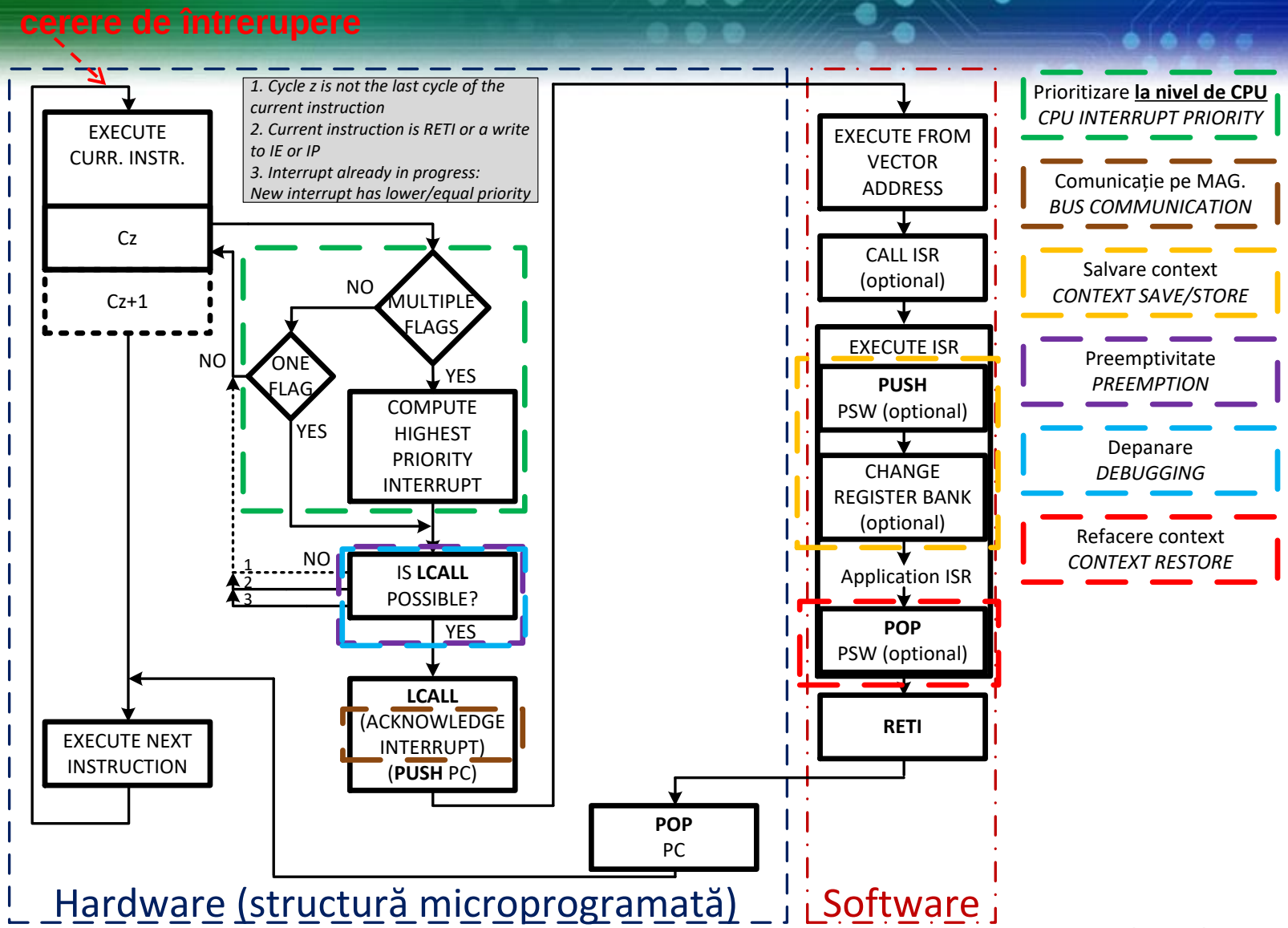
- Cazul întreruperilor la fel de prioritare
  - rezolvat printr-un mecanism intern de interogare

Ordine	SURSĂ		PRIORITATE
1	IE0	Întrerupere externă 0 (External Interrupt 0)	Cea mai mare
2	TF0	Cronometru 0 (Timer 0)	
3	IE1	Întrerupere externă 1 (External Interrupt 1)	
4	TF1	Cronometru 1 (Timer 1)	
5	RI + TI	Comunicație serială (Serial Port)	
6	TF2 + EXF2	Cronometru 2 (Timer 2) (doar 8052)	Cea mai mică

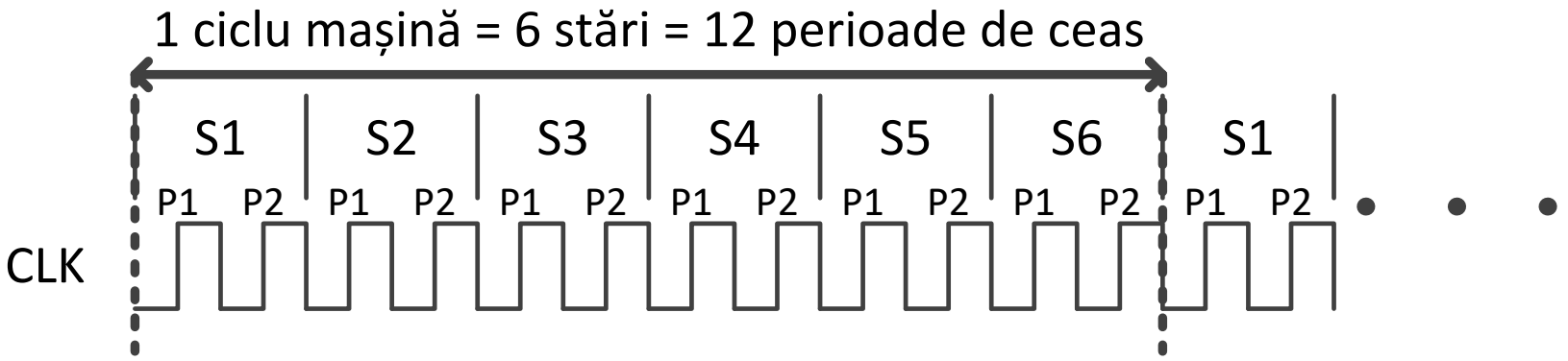
8051 Core – Algoritmul de întrerupere



8051 Core – Algoritmul de întrerupere

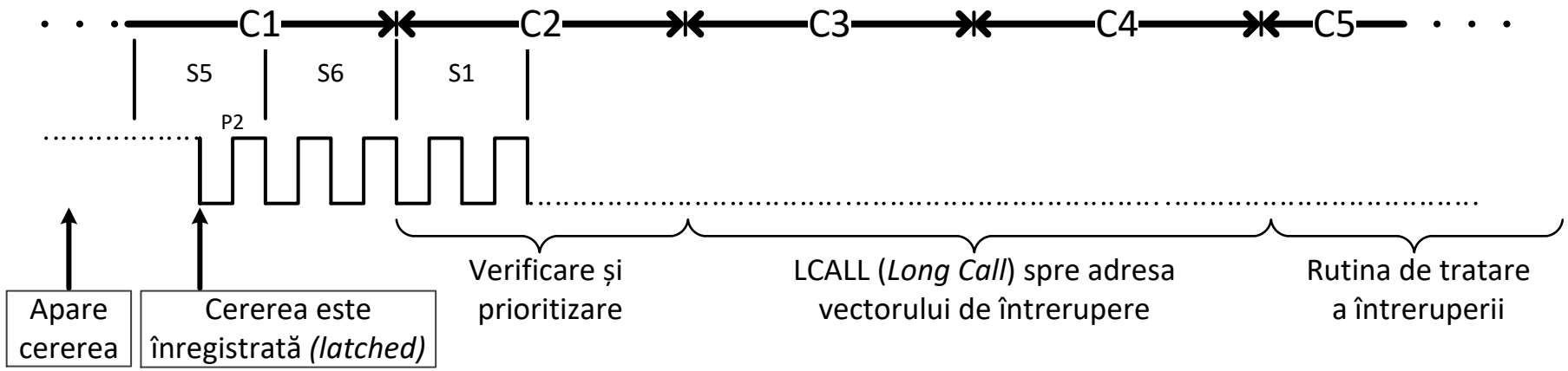


8051 Core – Analiza timpului



8051 Core – Analiza timpului

- Înregistrarea cererii: în S5P2
- Verificare (*Polling*) și prioritizare în caz de întreruperi “simultane”: în următorul ciclu
- T_{pro} = 3 x12 perioade + salvarea microprogramată a contextului



8051 Core – Latența întreruperii (nepreemptiv)

- Cel mai scurt timp posibil:
  - **Cererea apare la *sfârșitul* S5P1**
    - 3 (C1) + 12 x3 (C2, C3, C4) = 39 perioade de CLK
- Cel mai lung timp posibil:
  - **Cererea apare la *începutul* S5P2**
    - 3 *(C0)* + 12 x4 (C1, C2, C3, C4) = 51 perioade de CLK
  - **Cererea apare la *începutul* S5P2 + întârzieri din cauza instrucțiunii curente (MUL, DIV, RETI, acces la IE sau IP)**
    - 3 *(C0)* + 12 x8 = 99 perioade de CLK

8051 Core – Referințe

- Intel MCS51 Microcontroller Family User's Manual, 1994
- http://web.mit.edu/6.115/www/document/8051.pdf
- https://en.wikipedia.org/wiki/Intel_MCS-51

Arhitecturile
ARMv4, ARMv5, ARMv6

Arhitectura ARMv5

- 1993, Harvard sau von Neumann, ARM RISC, 51 instrucțiuni
- Folosește noțiunea de: “Excepție” (*Exception*)
- 7 tipuri de excepții
- 7 moduri de execuție (*Processor mode*)
  - **Modul utilizator: acces limitat la resurse**
  - **6 moduri de execuție privilegiată**
    - System mode – OS tasks
    - 5 *exception modes*

ARMv5 – Moduri de exec. (*Processor modes*)

Mod de execuție	Abre-viere	Codificare binară	Descriere și utilizare
User	usr	1 0000	Execuție normală a programului
FIQ	fiq	1 0001	Viteză mărită la: transferul de date / tratarea întreruperilor
IRQ	irq	1 0010	Tratare întreruperi de uz general
Supervisor	svc	1 0011	Mod protejat folosit de sistemul de operare
Abort	abt	1 0111	Implementare memorie virtuală și/sau mecanism de restricționare a accesului la memorie.
Undefined	und	1 1011	Simularea software a coprocesoarelor hardware
System	sys	1 1111	Execuția task-urilor sistemului de operare în mod privilegiat (drepturi depline de acces la resurse)

ARMv5 – *Privileged modes*

Mod de execuție	Abre-viere	Codificare binară	Descriere și utilizare
User	usr	1 0000	Execuție normală a programului
FIQ	fiq	1 0001	Viteză mărită la: transferul de date / tratarea întreruperilor
IRQ	irq	1 0010	Tratare întreruperi de uz general
Supervisor	svc	1 0011	Mod protejat folosit de sistemul de operare
Abort	abt	1 0111	Implementare memorie virtuală și/sau mecanism de restricționare a accesului la memorie.
Undefined	und	1 1011	Simularea software a coprocesoarelor hardware
System	sys	1 1111	Execuția task-urilor sistemului de operare în mod privilegiat (drepturi depline de acces la resurse)

ARMv5 – *Exception Modes*

Mod de execuție	Abre-viere	Codificare binară	Descriere și utilizare
User	usr	1 0000	Execuție normală a programului
FIQ	fiq	1 0001	Viteză mărită la: transferul de date / tratarea întreruperilor
IRQ	irq	1 0010	Tratare întreruperi de uz general
Supervisor	svc	1 0011	Mod protejat folosit de sistemul de operare
Abort	abt	1 0111	Implementare memorie virtuală și/sau mecanism de restricționare a accesului la memorie.
Undefined	und	1 1011	Simularea software a coprocesoarelor hardware
System	sys	1 1111	Execuția task-urilor sistemului de operare în mod privilegiat (drepturi depline de acces la resurse)

ARMv5 – Tabela vectorilor de întrerupere

Excepție	Mod procesor	Offset în tabela vectorilor
Reset	SVC	+0x00
Instrucțiune nedefinită	UND	+0x04
Software Interrupt (SWI)	SVC	+0x08
Prefetch Abort	ABT	+0x0C
Data Abort	ABT	+0x10
Rezervat	---	+0x14
IRQ (Interrupt)	IRQ	+0x18
FIQ (Fast Interrupt)	FIQ	+0x1C

ARMv5 – Vectori de întrerupere – implementare

0xXXXX 0000: 0xE59FFA38	**RESET**:	ldr	pc, [pc, *#reset*]
0xXXXX 0004: 0xEA000502	**UNDEF**:	b	*instrNedefinită*
0xXXXX 0008: 0xE59FFA38	**SWI**:	ldr	pc, [pc, *#swi*]
0xXXXX 000C: 0xE59FFA38	**PABT**:	ldr	pc, [pc, *#prefetch*]
0xXXXX 0010: 0xE59FFA38	**DABT**:	ldr	pc, [pc, *#data*]
0xXXXX 0014: 0xE59FFA38	**---**:	ldr	pc, [pc, *#rezervat*]
0xXXXX 0018: 0xE59FFA38	**IRQ**:	ldr	pc, [pc, *#irq*]
0xXXXX 001C: 0xE59FFA38	**FIQ**:	ldr	pc, [pc, *#fiq*]

Notă:
Tabela poate începe de la **0x0000 0000** sau de la **0xFFFF 0000**
(extragere instrucțiuni din FLASH – lentă – sau din SRAM – rapidă)

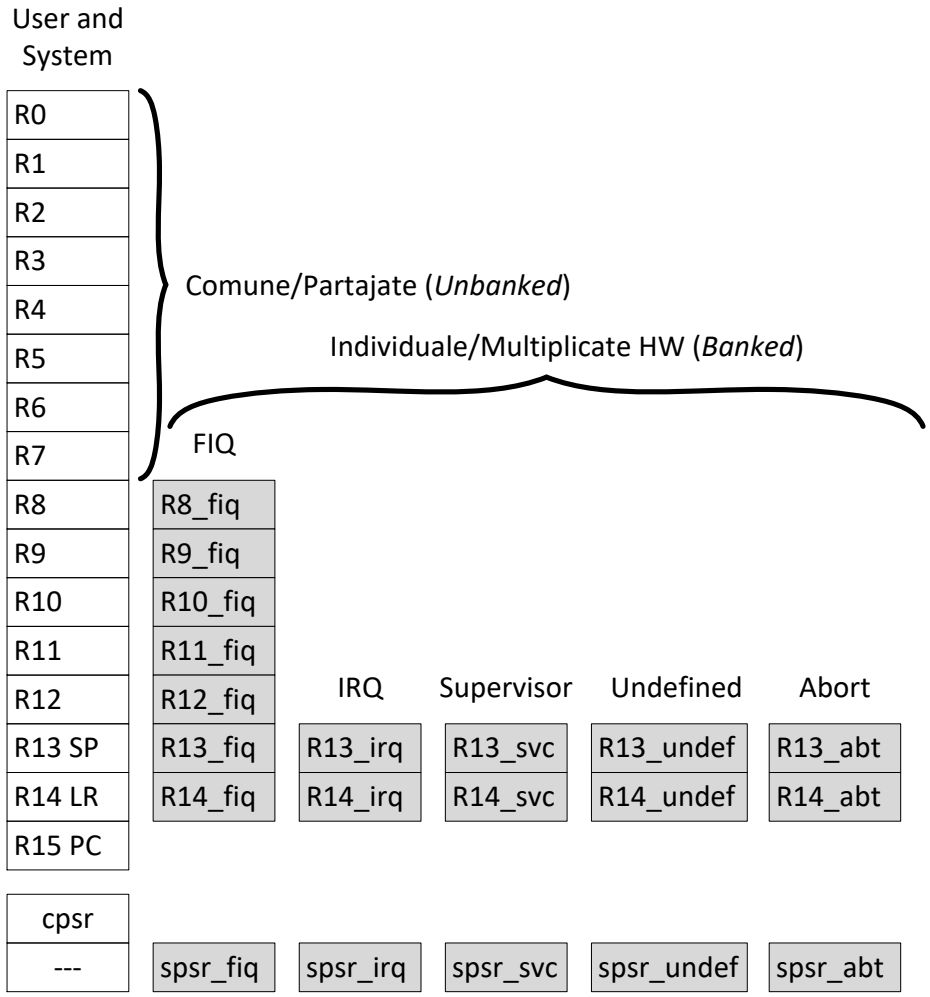
ARMv5 – Prioritatea excepțiilor

- Bit *I* = 1: IRQ disable
- Bit *F* = 1: FIQ disable
- FIQ , IRQ – active pe palier sau front

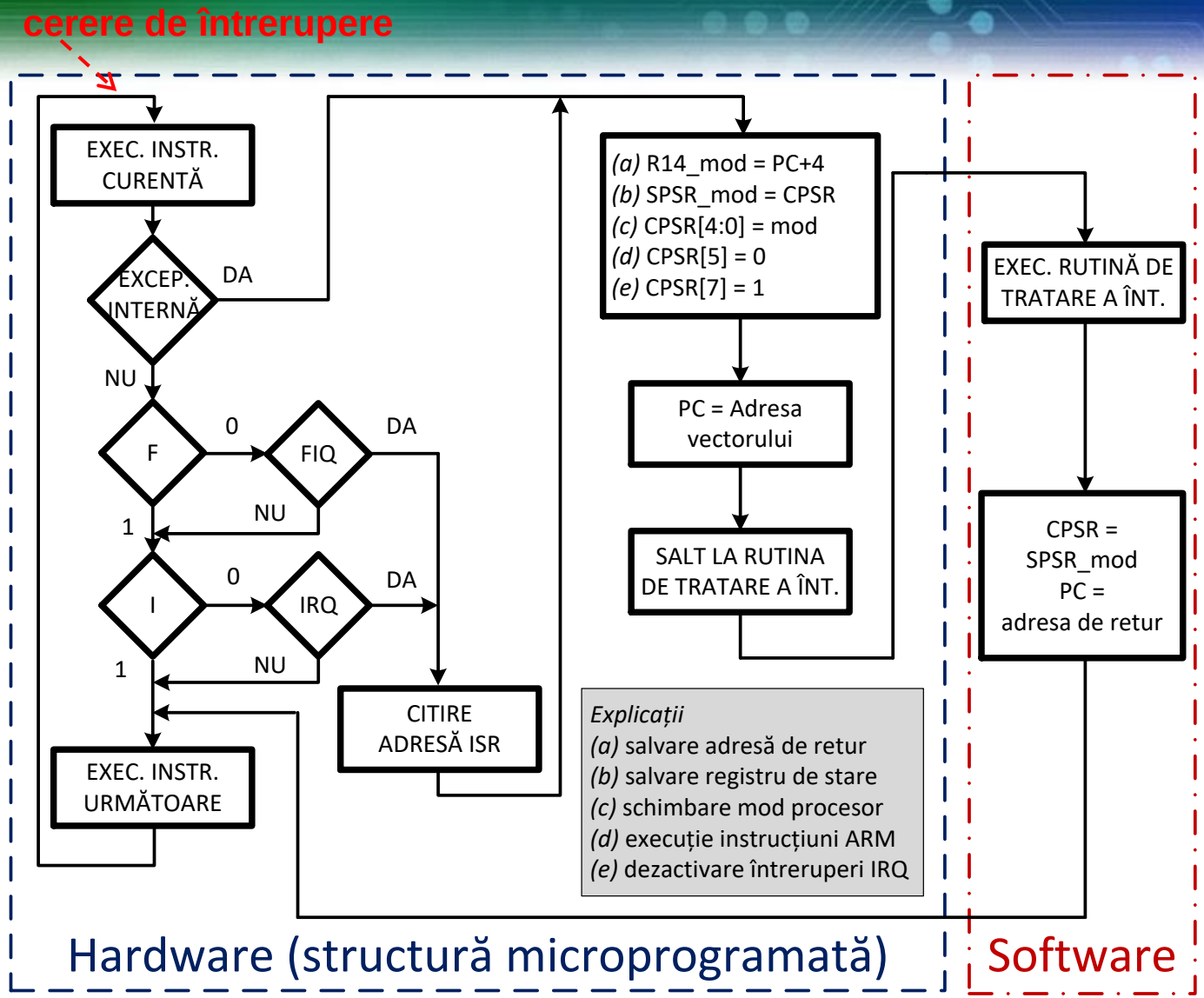
Numele excepției	Prio-ritate	Bit I	Bit F
Reset	1	1	1
Data Abort	2	1	--
Fast Interrupt Request	3	1	1
Interrupt Request	4	1	--
Prefetch Abort	5	1	--
Software Interrupt	6	1	--
Undefined Instruction	6	1	--

ARMv5 – Registrele CPU (în contextul excepțiilor)

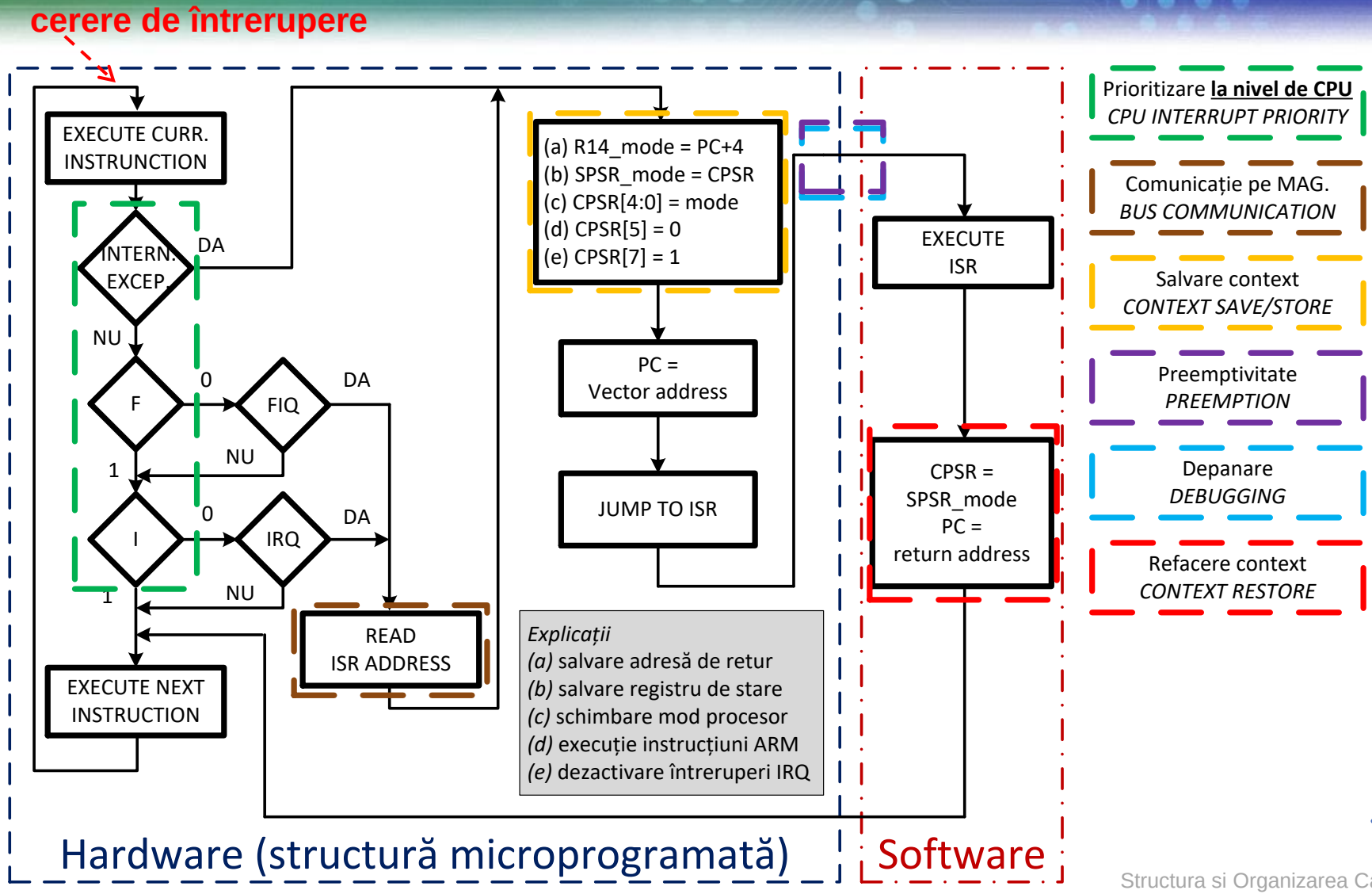
- R0-R7
  - **accesibile din orice mod de execuție**
- Rn, Rn_*mod*
  - **registre diferite**
    - locații de memorie diferite
- Registrul de tip spsr_*mod*
  - ***saved program status***
  - **un registru dedicat fiecărui mod de execuție privilegiată**



ARMv5 – Algoritmul excepțiilor



ARMv5 – Algoritmul excepțiilor



Prioritizare **la nivel de CPU**
CPU INTERRUPT PRIORITY

Comunicație pe MAG.
BUS COMMUNICATION

Salvare context
CONTEXT SAVE/STORE

Preemptivitate
PREEMPTION

Depanare
DEBUGGING

Refacere context
CONTEXT RESTORE

ARMv5 – Analiza timpului

- *Low latency interrupt configuration*
  - ***specific implementării:***
    - existența mecanismului
    - funcționarea mecanismului
    - efecte asupra memoriei/perifericelor procesorului
  - **Instrucțiuni *multi-access* intreruptibile**

ARMv5 – Analiza timpului

- Timpul maxim pt. ca cererea să fie detectată
  - **Tsyncmax = 4 perioade CLK**
- Timp de exec. al instrucțiunii LDM (*Load Multiple*)
  - **Tldm = 20 perioade CLK**
- Timp de intrare în Data Abort
  - **Texc = 3 perioade CLK**
- Timp de intrare în FIQ
  - **Tfiq = 2 perioade CLK**

ARMv5 – Latența întrereuperii (nepreemptiv)

- Cel mai scurt timp posibil
  - **Tsyncmin + Tfiq = 5 perioade CLK**
- Cel mai lung timp posibil
  - **Tsyncmax + Tldm + Texc + Tfiq = 29 perioade CLK**
- Dupa acest timp, începe execuția de la +0x1c (FIQ)

ARMv5 – Referințe

- ARM DDI 0100I - ARM Architecture Reference Manual, 2005
- ARM DDI 0210C - ARM7 TDMI Technical Reference Manual, R4p1, 2004
- ARM System Developer's Guide – Designing and Optimizing System Software, A. N. Sloss, D. Symes, C. Wright, Elsevier MK, 2004
- UM10237 – NXP LPC24XX User manual, Rev. 04 – 2009
- https://developer.arm.com/ip-products/processors/classic-processors

Implementări folosind ARM7TDMI
Nokia 6110, 3210, 3310

- Chiar și în anul 2017, ARM7TDMI era cel mai vândut MCU ARM (aprox. 90buc./secundă)

- https://en.wikipedia.org/wiki/ARM7#ARM7TDMI
- https://en.wikipedia.org/wiki/Nokia_6110
- https://en.wikipedia.org/wiki/Nokia_3210
- https://en.wikipedia.org/wiki/Nokia_3310

Arhitectura Cortex ARMv8-M

Arhitectura ARMv8-M

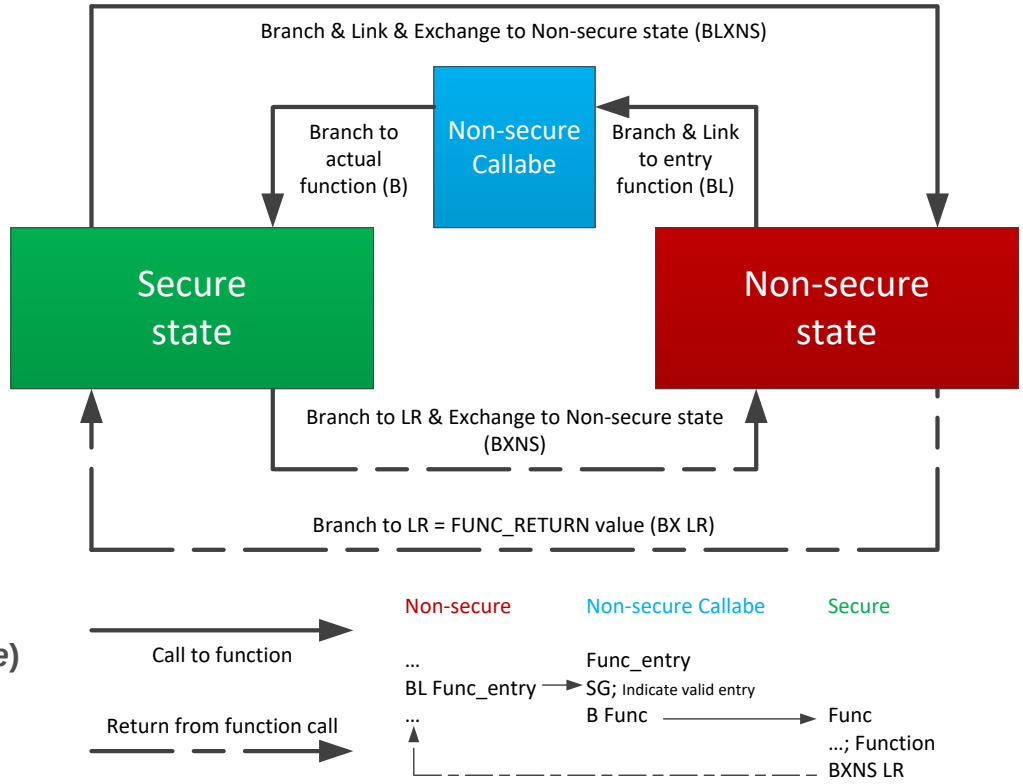
- 2016, von Neumann, ARM RISC
- ARMv8-M Baseline
  - similar cu ARMv6-**M**, îmbunătățită
  - set de instrucțiuni redus (M23: 83 instrucțiuni)
  - număr redus de porți (între 12k (M0+) și 24k (M3))
  - ideal pentru consum redus de energie
- ARMv8-M Mainline
  - similar cu ARMv7-**M**, îmbunătățită
  - set de instrucțiuni pentru procesare complexă de date (M33: ≈300)
    - Ex: operații pe vectori de date (SIMD); o instrucțiune = mai multe operații
  - ideal pentru sisteme de calcul de înaltă performanță
- Nu conține registre multiplicate (*register banks*)
  - mai puțină suprafață de siliciu și mai puțin consum de energie

Arhitectura ARMv8-M – *TrustZone*

- Tehnologia TrustZone – securitatea execuției
  - **conceptual** similar cu TrustZone din arhitectura Cortex-A
  - optimizat pentru sisteme incorporate de timp real
  - optimizat pentru consum redus de energie
  - **implementare opțională (de către proiectanții de MCU / SoC)**
- Întrerupere *Secure* sau *Non-Secure*
  - configurabil doar din modul de execuție securizat (*Secure)*
  - nu există restricții privind tratarea în funcție de mod
  - **acces diferențial la resurse / zone de memorie**

Arhitectura ARMv8-M – *TrustZone*

- B
  - **Salt (*branch*)**
- BL
  - **Salt și creare de legătură de întoarcere (*branch & link*)**
- BX
  - **Salt și schimbare către starea securizată (*branch & exchange to secure state*)**
- BXNS
  - **salt și schimbare către starea nesecurizată (*branch & exchange to non-secure state*)**
- BLXNS
  - **BL și schimbare către starea nesecurizată (*branch & link & exchange to non-secure state*)**



ARMv8-M
Baseline

- Prioritate *negativă*
  - fixă (cablată)
- Prioritate configurabilă
  - pe 4 niveluri

Excepție	ID	Prioritate
Reset	1	-4 (maximă)
Secure HardFault (AIRCR.BFHFNMINS = 1)	3	-3
NMI	2	-2
Secure HardFault (AIRCR.BFHFNMINS = 0)	3	-1
Non-Secure HardFault	3	-1
Reserved	4-10	---
SVCall	11	Configurabil
Reserved	12-13	---
PendSV	14	Configurabil
SysTick	15	Configurabil
External Interrupt 0	16	Configurabil
…	…	Configurabil
External Interrupt N (N < 496)	16 + N	Configurabil

ARMv8-M
Mainline

- În plus față de Baseline
  - excepții pentru:
    - **securitatea** aplicației
    - **siguranța** în funcționare
  - 8 – 256 niveluri de prioritate
    - numărul de niveluri depinde de implementare

Excepție	ID	Prioritate
Reset	1	-4 (maximă)
Secure HardFault (AIRCR.BFHFNMINS = 1)	3	-3
NMI	2	-2
Secure HardFault (AIRCR.BFHFNMINS = 0)	3	-1
Non-Secure HardFault	3	-1
MemManage fault	4	Configurabil
BusFault	5	Configurabil
UsageFault	6	Configurabil
SecureFault	7	Configurabil
Reserved	8-10	---
SVCall	11	Configurabil
DebugMonitor	12	Configurabil
Reserved	13	---
PendSV	14	Configurabil
SysTick	15	Configurabil
External Interrupt 0	16	Configurabil
…	…	Configurabil
External Interrupt N (N < 496)	16 + N	Configurabil

ARM Cortex M23
MCU Core

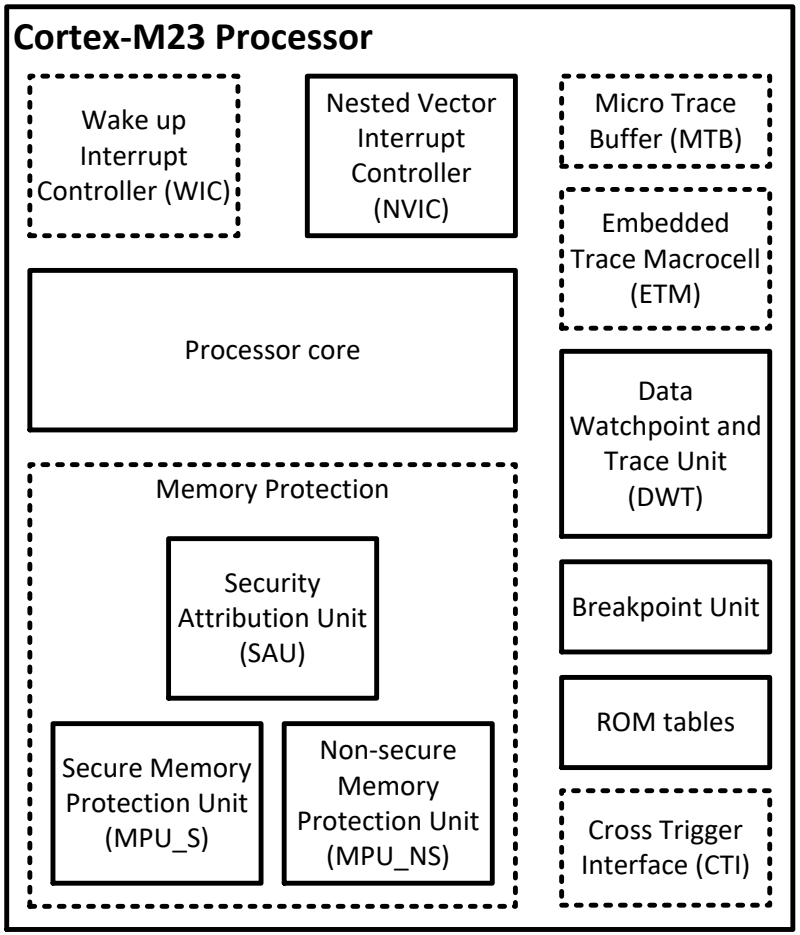
ARMv8-M – Cortex M23

- O implementare a arhitecturii ARMv8-M Baseline

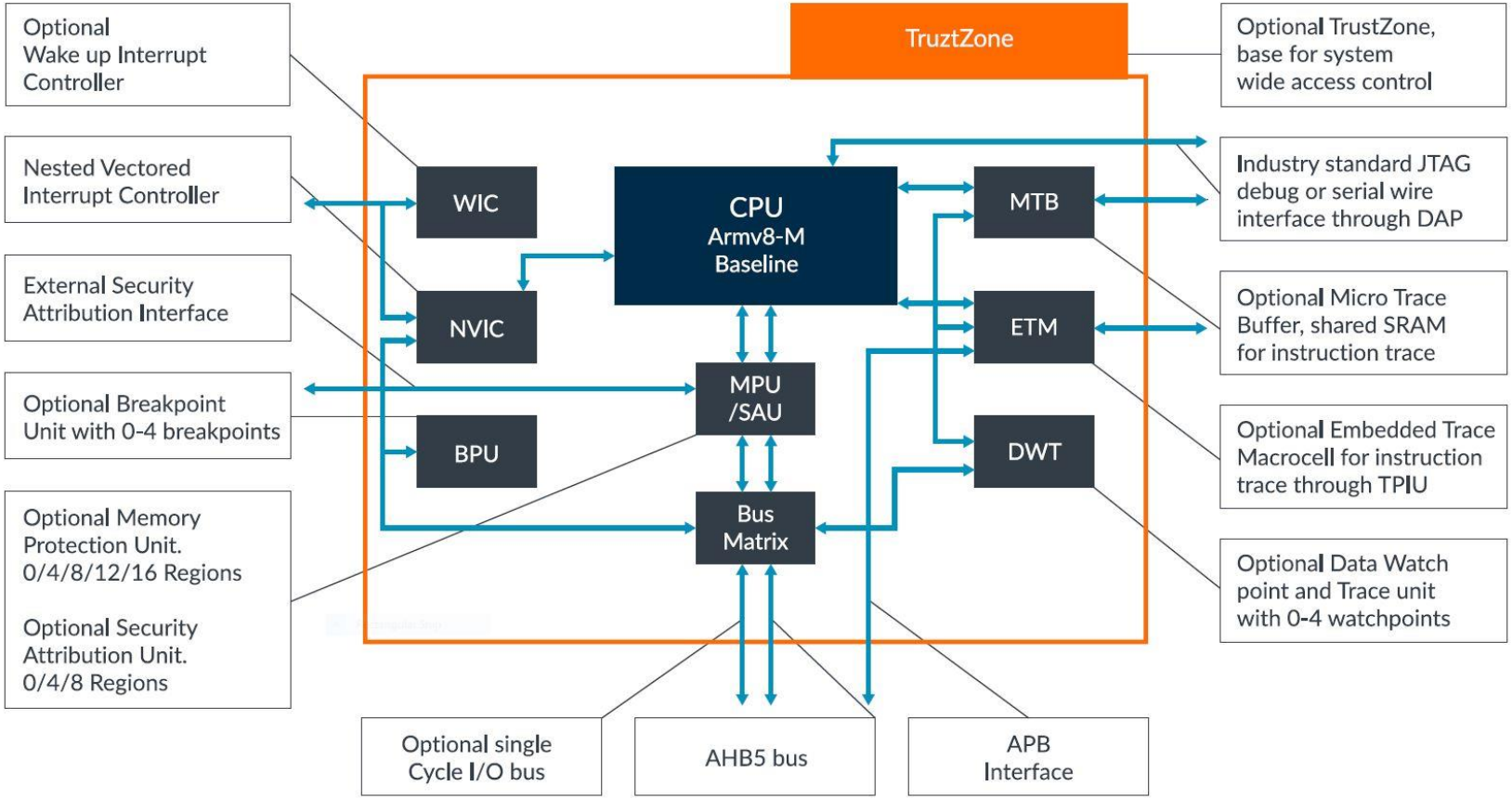
- **NVIC încapsulat alături de Core**
  - viteză mărită de comunicație

- Wake up Interrupt Controller
  - conceput pentru îmbunătățirea echilibrului între un consum redus de energie (*sleep*) și timpul de răspuns/reacție specific aplicațiilor de tip **Real-Time**

implementare opțională (la alegerea integratorului)



ARMv8-M – Cortex M23



ARMv8-M – Cortex M23 – Excepții

- Starea unei excepții:
  - **inactivă (cerere inexistentă) (*Inactive)***
  - **în așteptare (spre a fi tratată) (*Pending)***
  - **activă (este în curs de tratare) (*Active*)**
    - în caz de preemptivitate, ambele(!) excepții sunt active
  - **activă și în așteptare (*Pending & Active)***
    - o excepție este în curs de tratare, iar o altă cerere pentru același tip de excepție este în așteptare

ARMv8-M – Cortex M23 – Excepții

- Tipuri de excepții:
  - Reset
  - NMI
  - HardFault
  - Secure HardFault
  - SVCall
  - PendSV
  - SysTick
  - Interrupt (IRQ)

ARMv8-M – Cortex M23 – Excepții

- Categorii de tratare a excepțiilor
  - **ISRs**
    - IRQ0 – IRQ239
      - *Secure / Not Secure*
  - **Fault**
    - HardFault
      - *Secure / Not Secure*
  - **System**
    - NMI, PendSV, SVCall, Systick, HardFault
      - *Secure / Not Secure*

ARMv8-M – Cortex M23 – Vectori de excepție

- După resetarea sistemului, tabela începe la adresa 0x00000000
  - Tabela poate fi relocată apoi prin setarea registrului VTOR
- Configurabil:
  - NMI și HardFault
    - prezente în ambele stări (securizată, nesecurizată)

    sau
    - prezente doar în starea securizată

*Doar în cazul implementării extensiei de securitate

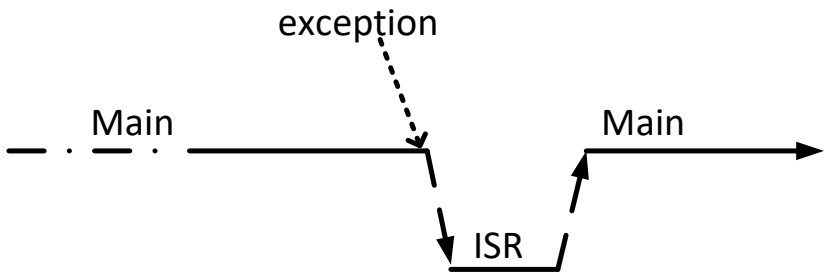
Nr. Exepție	Nr. ÎNT.	Secure Vector*	Non-secure Vector	Offset
		Valoare inițială a Pointerului de Stivă (SP)		0x00
1		Reset		0x04
2		NMI_S	NMI_NS	0x08
3		HardFault_S	HardFault_NS	0x0C
4..10		Rezervat	Rezervat	…
11		SVCall_S	SVCall_NS	0x2C
12		-	-	-
13		Rezervat	Rezervat	0x30
14		PendSV_S	PendSV_NS	0x38
15		SysTick_S	SysTick_NS	0x3C
16	0	IRQ0	IRQ0	0x40
17	1	IRQ1	IRQ1	0x44
18	2	IRQ2	IRQ2	0x48
…	…	…	…	…
255	239	IRQ239	IRQ239	0xBC

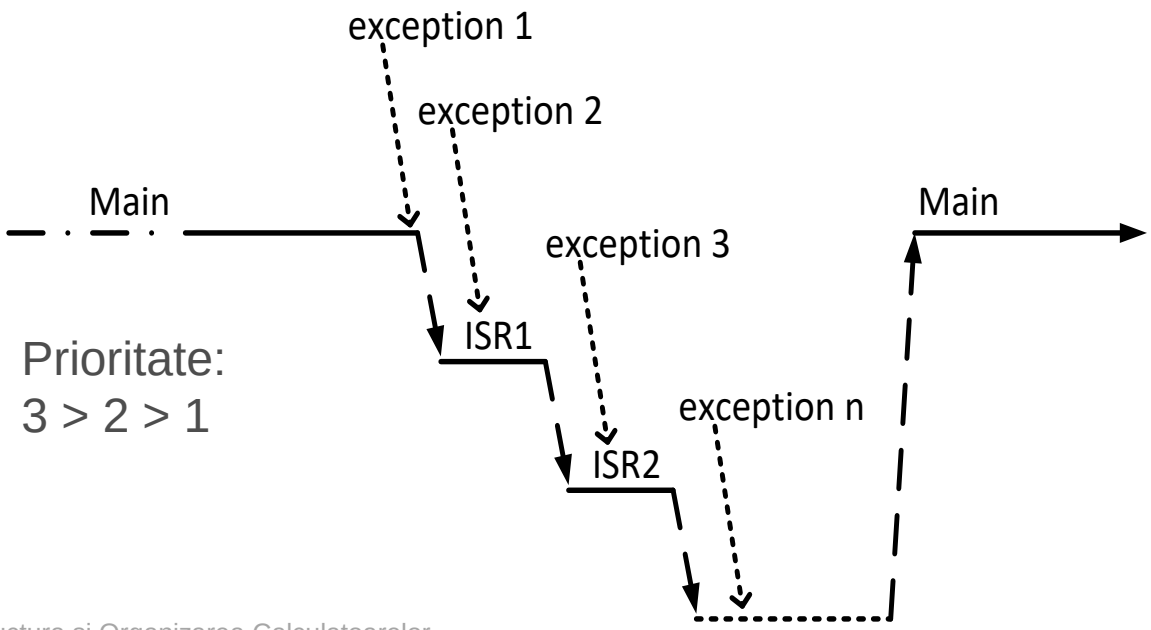
ARMv8-M – Cortex M23 – Excepții

- Metode de tratare a excepțiilor:
  - **clasic (nepreemptiv)**
  - **preemptiv (*nested exceptions*)**
    - limita dată de stivă
    - n să nu depășească 3 sau 4 niveluri
      - 1-2 kernel
      - 3-4 aplicatie

      (good practice)

n_{max}=?



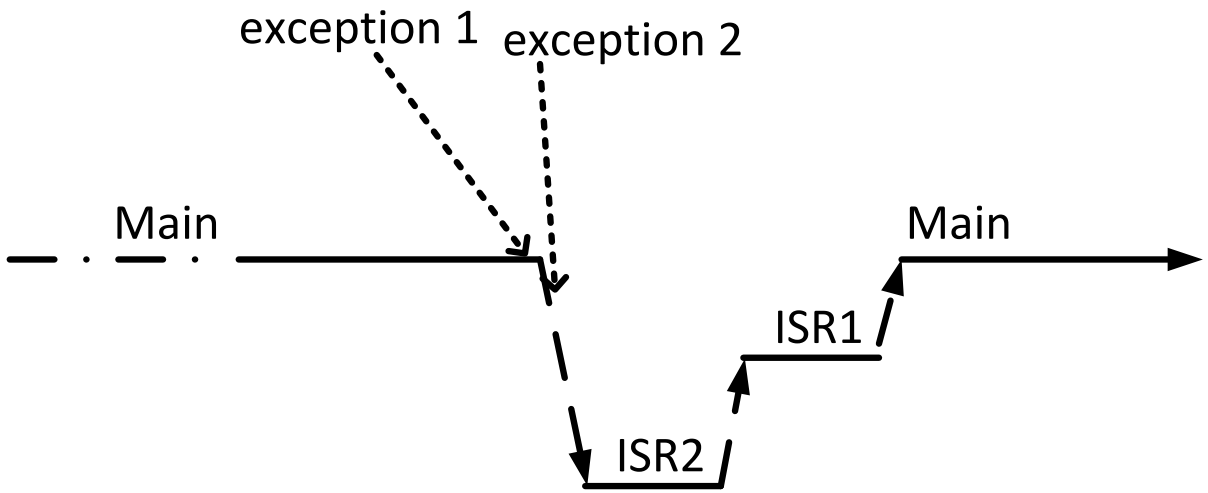


Prioritate:
3 > 2 > 1

ARMv8-M – Cortex M23 – Excepții

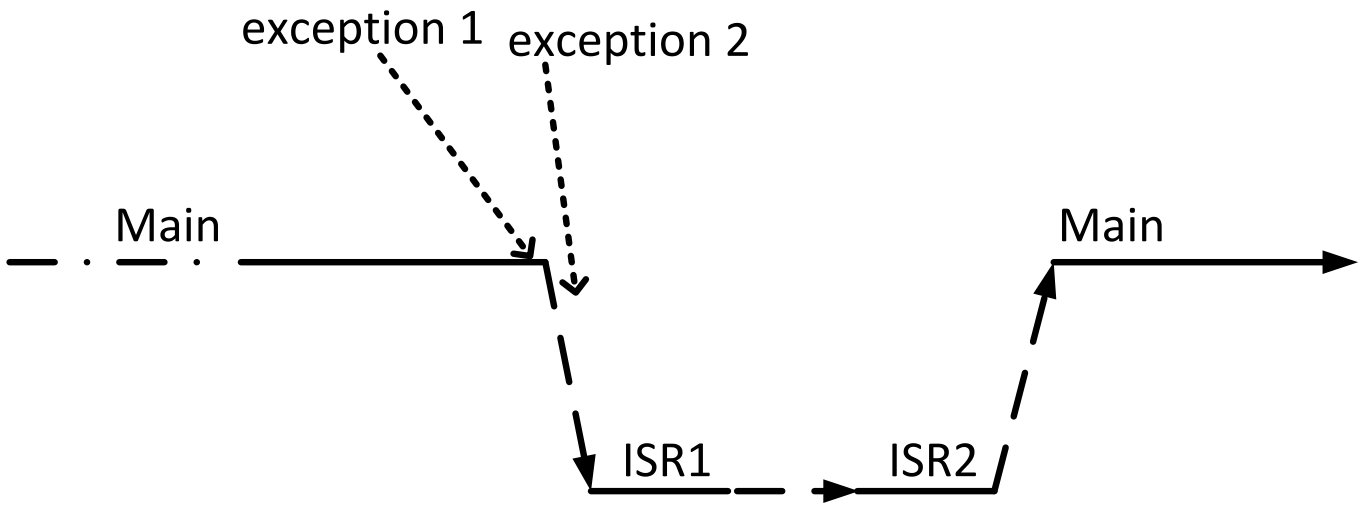
- Metode de tratare a excepțiilor:
  - **cu sosire întârziată (*late arrival exception*)**
    - în cazul în care o altă excepție mai prioritară survine în timpul operației de salvare de context
    - tratarea noii excepții se face imediat după operația de salvare de context inițiată pentru excepția anterioară
      - îmbunătățește viteza de reacție în caz de preemptivitate

Prioritate:
2 > 1



ARMv8-M – Cortex M23 – Excepții

- Metode de tratare a excepțiilor:
  - **înlănțuită (*linked*)**
    - În anumite condiții, transferul cu stiva se omite, deoarece contextul la care trebuie sa se revină este același

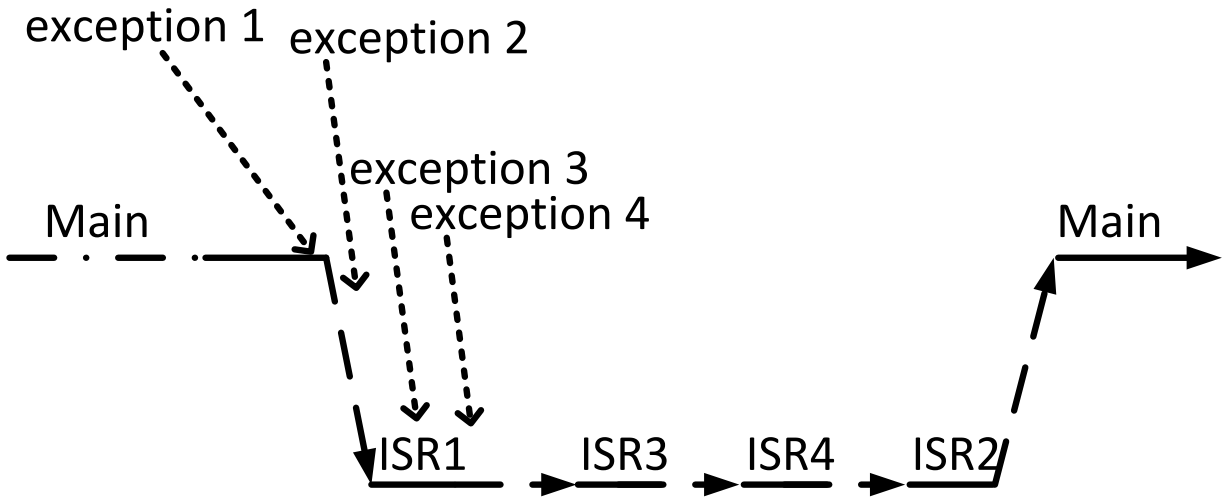


Prioritate:
1 >= 2

ARMv8-M – Cortex M23 – Excepții

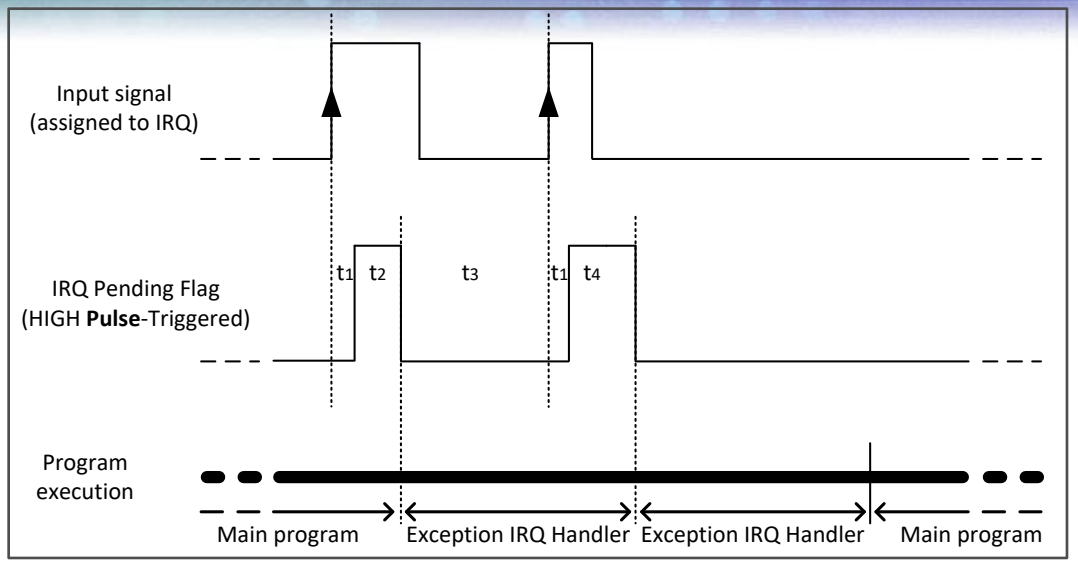
- Cazul întreruperilor la fel de prioritare, în starea de așteptare

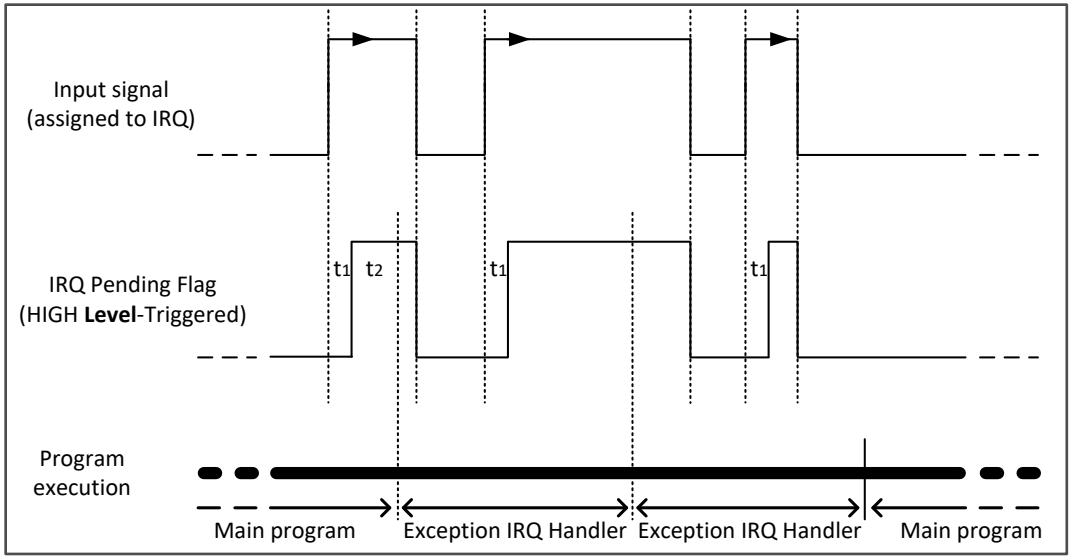
Nume	Offset în tabela vectorilor	Premise
Exception 1	irelevant	Offset: a < b < c Prioritate: e1 >= e2 = e3 = e4
Exception 2	c	
Exception 3	a	
Exception 4	b	



ARMv8-M – Cortex M23 – Excepții

- t_1 – perioada de timp necesară detecției și memorării (*latching)*
- t_2 – perioada de timp formată din timpul de:
  - **terminare/întrerupere a instrucțiunii curente**
  - **salvare context**
  - **schimbare de context**
- t_3 – perioada de timp până la următorul impuls
- t_4 – perioada de timp necesară pentru terminarea execuției rutinei de tratare a întreruperii





ARMv8-M – Cortex M23 – Excepții – intrare

- Condiții:
  - **există excepție în starea de așteptare (*Pending*)**
  - **procesorul rulează în modul Thread**
  - **excepția mai recentă are o prioritate mai mare decât excepția în curs (preemptivitate)**

- La intrare:
  - **stocarea informațiilor pe stivă (*stacking*)**
    - cadrul de date stocat se numește *stack frame*
      - cadru scurt (*short stack frame*)
      - cadru extins (*extended stack frame*)

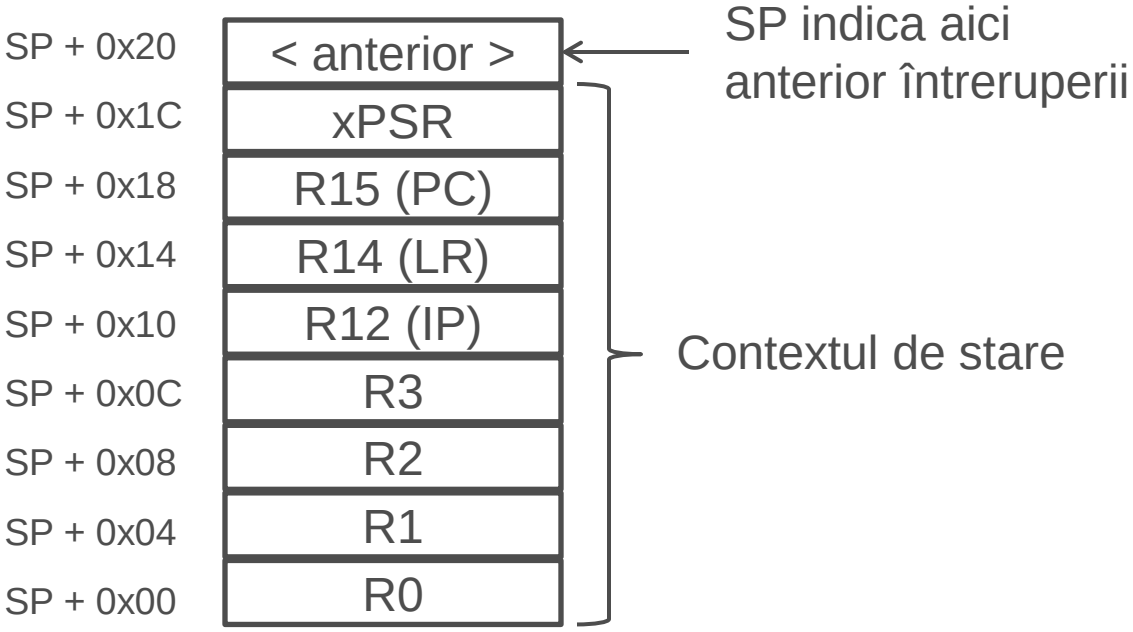
ARMv8-M – Cortex M23 – Stivă – cadrul scurt

R0 – R3: registre de lucru (folosite în instrucțiuni)

IP – intra-procedure-call register: registru pentru transmisia de date între rutină și o subrutină de-a sa.

LR – Link Register – conține o valoare de tip **EXC_RETURN**

PC – Program Counter

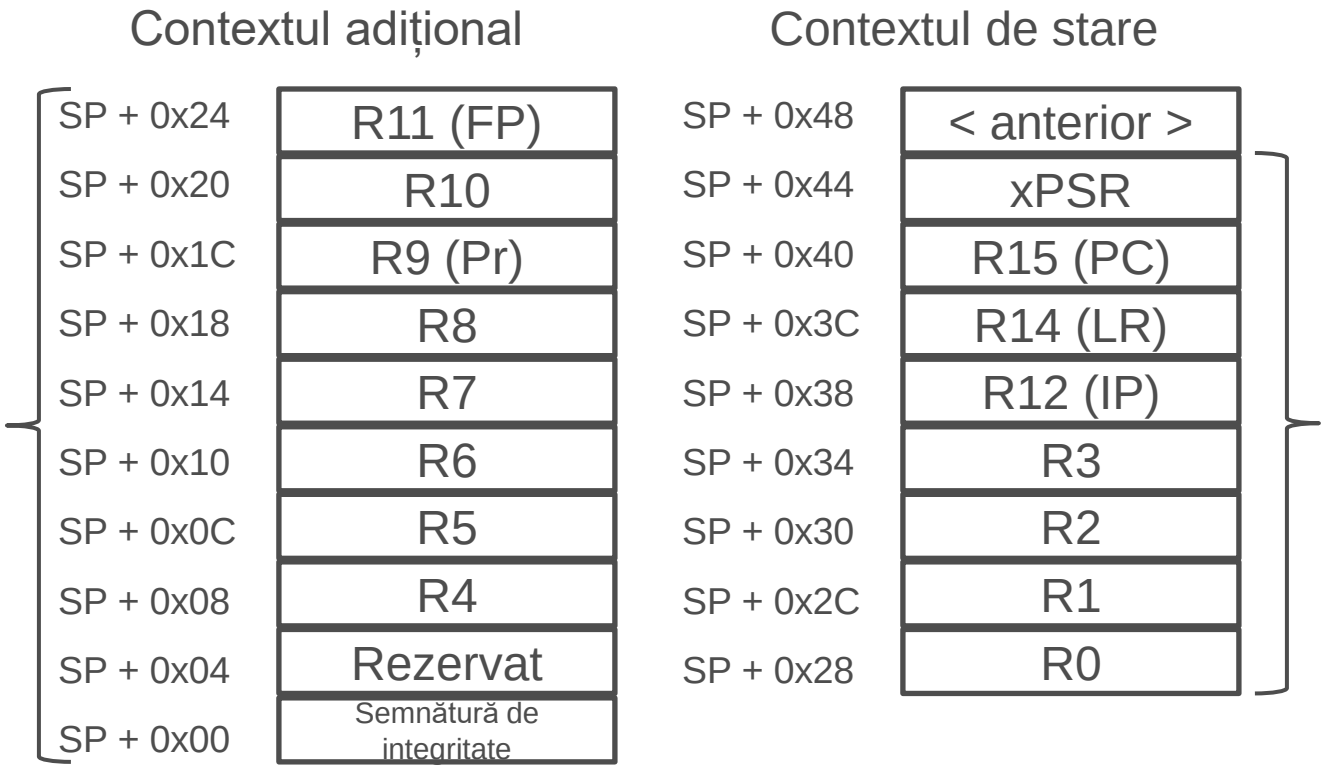


ARMv8-M – Cortex M23 – Stivă – cadrul extins

R4 – R8, R10, R9: folosite la stocarea variabilelor locale

Pr – Platform register – funcționalitate dependentă de platformă

FP – Frame Pointer – indică stiva folosită la întoarcere



Contextul aditional:
salvat atunci când se întrerupe execuția securizată, pentru a trata o excepție configurată ca fiind nesecurizată

ARMv8-M – Cortex M23 – Excepții - intrare

- Cadrul scurt este utilizat atunci când:
  - **intrarea se face din starea *Non-secure***
  - **sau opțiunea *Security extension* nu este implementată**
  - **sau *extended stack frame* nu este necesar**

- Cadrul extins este utilizat atunci când:
  - **o excepție nesecurizată (*Non-secure code*) întrerupe o excepție securizată (*Secure code*)**
  - **sau în cazul utilizării excepțiilor cu sosire întârziată și excepția finală este securizată**

ARMv8-M – Cortex M23 – Excepții - revenire

- Revenirea din tratarea excepției se realizează:
  - **când rutina de tratare s-a încheiat, și:**
    - nu există vreo altă excepție mai prioritară în așteptare
    - nu s-a tratat o excepție în modul de sosire întârziată
  - **prin actualizarea registrului PC prin:**
    - POP
    - BX reg
    - BX LR (*Link Register*)
      - *Link Register* conține informații pentru revenire, **nu** adresa de revenire. Informațiile sunt de forma *EXC_RETURN*

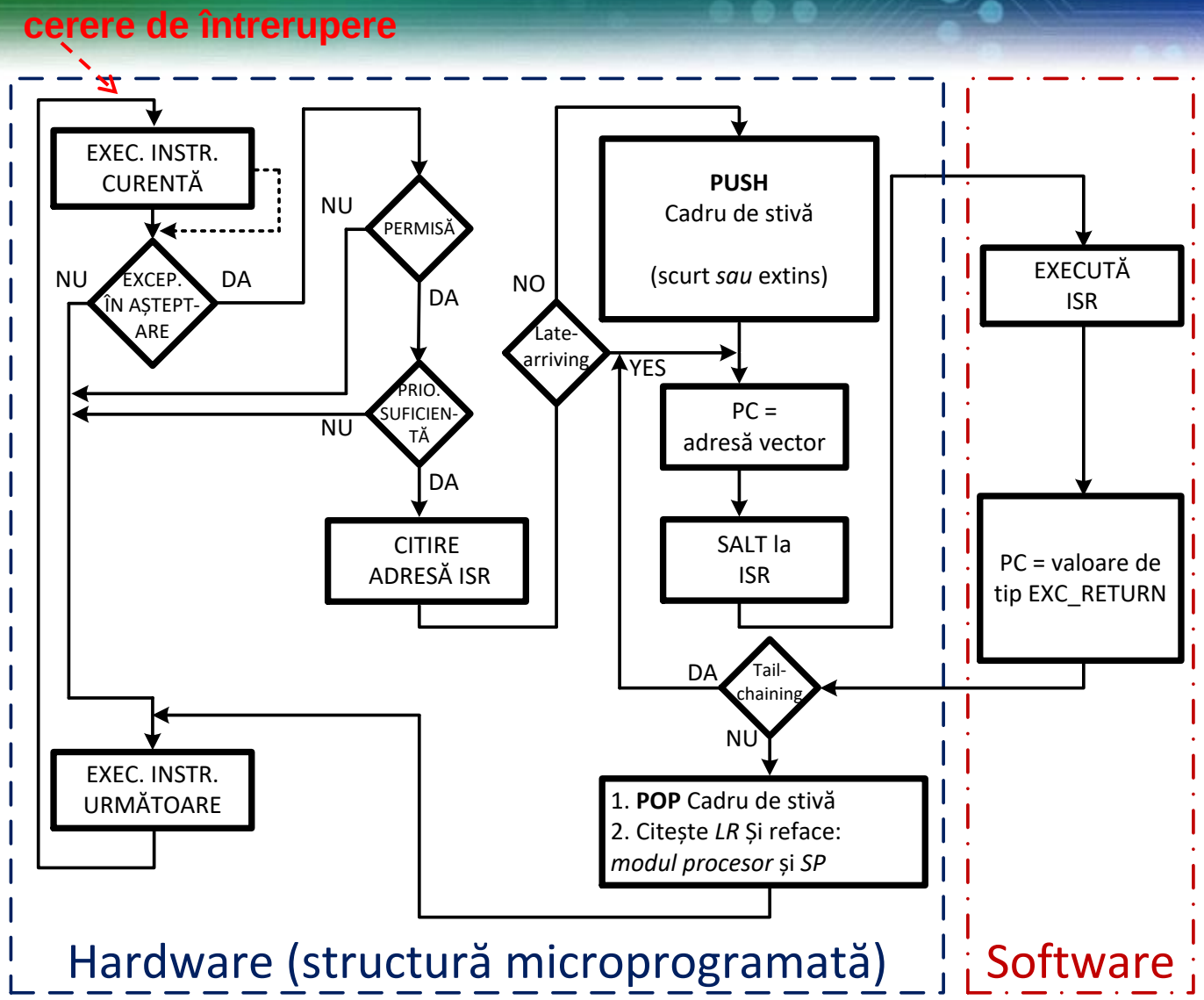
ARMv8-M – Cortex M23 – Excepții - revenire

Formatul valorii de tip EXC_RETURN

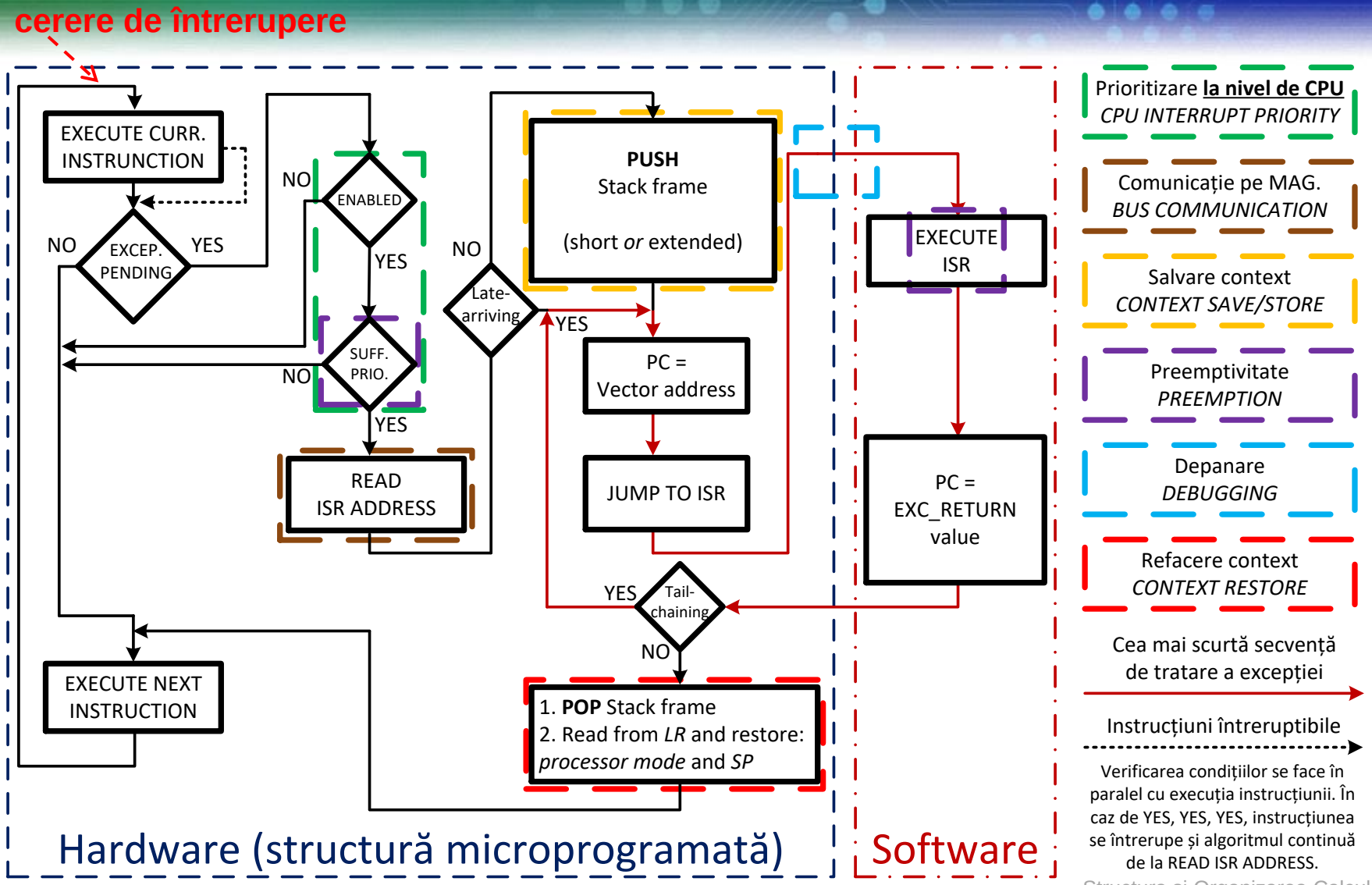
Biți	Denumire	Funcție
[31:24]	PREFIX	Identifică formatul de tip EXC_RETURN Se citește mereu 0b1111 1111
[23:7]	---	Rezervat (RES1)
[6]	S	Indică dacă s-a folosit o stivă securizată (1) sau nesecurizată (0). *= RES0
[5]	DCRS	Indică dacă e necesară salvarea contextului (1) sau nu este necesară (0). *= RES1
[4]	---	Rezervat (RES1)
[3]	Mode	Indică modul procesor în care s-a produs excepția. (0 = Handler, 1 = Thread)
[2]	SPSEL	Indică pointerul de stivă folosit de excepție (0 = Pointerul principal, 1 = Pointerul de proces)
[1]	---	Rezervat
[0]	ES	Indică starea de securitate unde se face tratarea excepției (0 = nesecurizată, 1 = securizată). *= RES0

* valoarea de reset, dacă extensia de securitate nu este implementată

ARMv8-M – Cortex M23 – Algoritmul excepției



ARMv8-M – Cortex M23 – Algoritmul excepției



ARMv8-M – Cortex M23 – Analiza timpului

- Pot fi suspendate cele mai lungi instrucțiuni :
  - **aritmetice**
    - înmulțire
    - împărțire
  - **de lucru cu memoria**
    - încărcare multiplă (*store-multiple*)
    - descărcare multiplă (*load-multiple*)
- La revenire, instrucțiunea suspendată se reia de la început
- Depinde de implementarea procesorului:
  - **dacă asigură eliminarea abaterii (*jitter*) pentru cea mai prioritară întrerupere**
- Reducerea abaterii se realizează și prin metodele *late-arrival* și *tail-chaining*

ARMv8-M – Cortex M23 – Analiza timpului

- În cazul nepreemptiv, fără *late-arriving*:
  - **Dacă extensia de securitate nu este implementată sau se utilizează cadrul de stivă scurt, latența întreruperii este de 15 perioade**
  - **Dacă extensia de securitate este implementată și se utilizeaza cadrul de stivă extins, latența întreruperii este de 27 perioade**

- Latența a fost calculată ca numărul de perioade din momentul apariției întreruperii până la ciclul de acces la date al primei instrucțiuni din ISR.

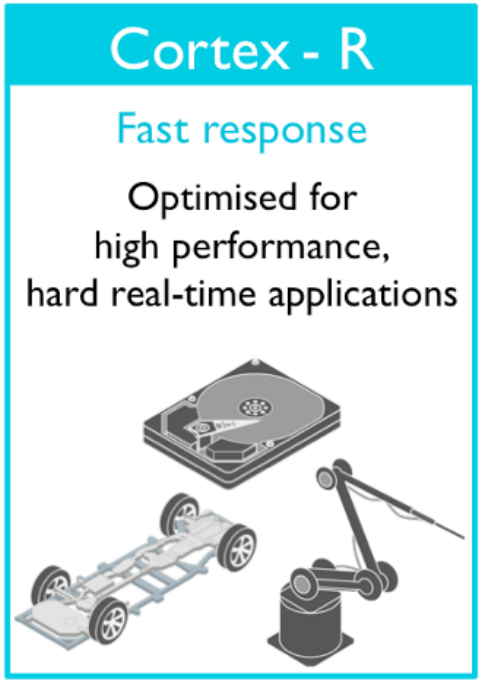
- Calculul exclude eventualitatea vreunei erori (*Fault*)

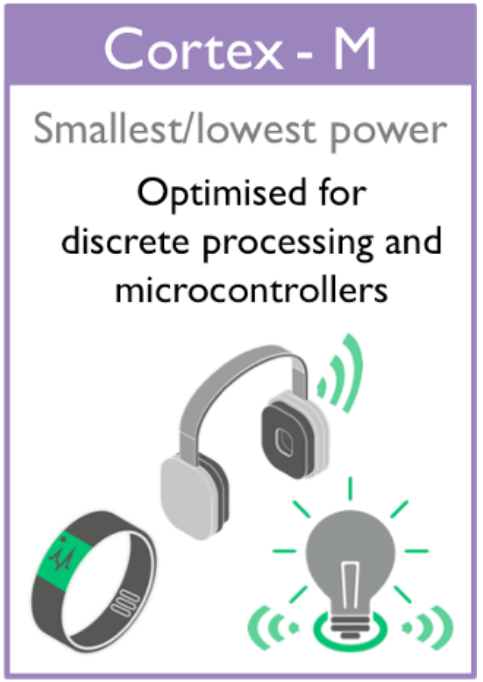
ARMv8-M – Referințe

- ARMv8-M Architecture Reference Manual, 12th release, **30.06.2020**
- ARM Cortex-M23 Devices Generic User Guide, DUI 1095A, rev. r1p0, 2018
- ARM Cortex-M23 Processor Technical Reference Manual, DDI 0550C, rev. r1p0, 2016
- ARM Cortex-M23 Processor Datasheet, 2020
- Procedure Call Standard for the Arm Architecture, IHI 0042J, Release 2020Q2, Jun 2020
- https://developer.arm.com/ip-products/processors/cortex-m/cortex-m23
- https://developer.arm.com/documentation/100690/0200

ARM Cortex – Implementări posibile

- https://community.arm.com





Comparația abaterilor

- Abaterea (*Jitter*)

  = (nr. maxim cicli – nr. minim cicli) / nr. maxim cicli

Procesor (Arhitectură)	Nr. max. cicli	Nr. min. cicli	Max. – min.	Jitter
8086	245	62	183	75%
8051	99	39	60	61%
ARM7 TDMI (ARMv5)	29	5	24	83%
Cortex M23 (ARMv8-M)	15 / 27 *	15 / 27 *	0	0%

* extensia de securitate nu este implementată **sau** se folosește cadrul scurt / extensia de securitate este implementată **și** se folosește cadrul extins

Concluzii – perspective actuale

- Tehnologia de integrare: chip extern VS încapsulare
  - **NVIC încapsulat alături de procesor**
  - **Controller de întreruperi extern încapsulat alături de Core, în MCU**
- Tehnologia schimbării și refacerii contextului de întrerupere
  - **Ocupare variabilă a stivei**
  - **Registre dedicate (LR, SP), de uz general și registrul de stare**
    - duplicate pentru contextul de execuție al întreruperii
  - **Mecanismul de preemptivitate**
    - tratarea înlănțuită a excepțiilor
    - sosire întârziată a excepțiilor
- Determinism perfectibil
  - **abaterea tinde spre 0%**
  - **execuția instrucțiunilor organizată sub formă de pipeline**
    - majoritatea instrucțiunilor executate într-un singur ciclu
    - posibilitate de întrerupere la fiecare ciclu procesor
  - **instrucțiuni multi-ciclu întreruptibile**
- Execuție securizată a rutinei de tratare a întreruperii

- Memoria -

- analiză comparativă -

Memoria

Compiler: Error at line 40

Me: "What? How? My code only has 30 lines

Compiler:

Memoria – Generalități

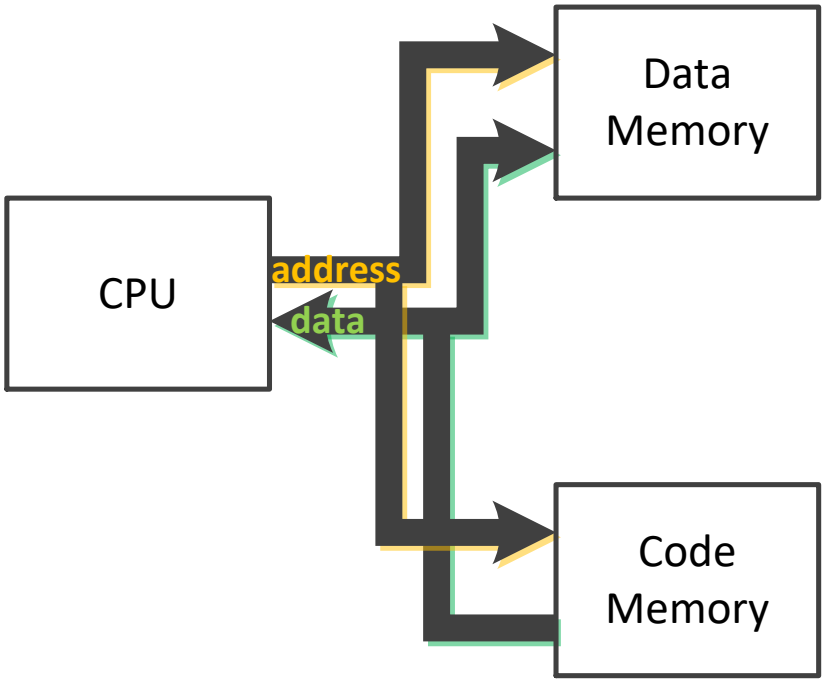
- Arhitecturi de calculatoare (dpdv. al interacțiunii CPU ↔ memorie)
  - **von Neumann** *(în literatură și sub denumirea: Princeton model)*
  - **Harvard**
  - **Advanced Harvard**
  - **hibrid: mixtură între von Neumann și Harvard (core Harvard, mem ext. von Neumann)**
    - *…filosofic vorbind. Practic, este fie von Neumann, fie Harvard.*

- Convenția de stocare (*endianess*)
  - **Little-endian** *(lit. Intel convention) – octetul cel mai semnificativ la adresa* ***mai mare***
  - **Big-endian** *(lit. Network order) – octetul cel mai semnificativ la adresa* ***mai mică***

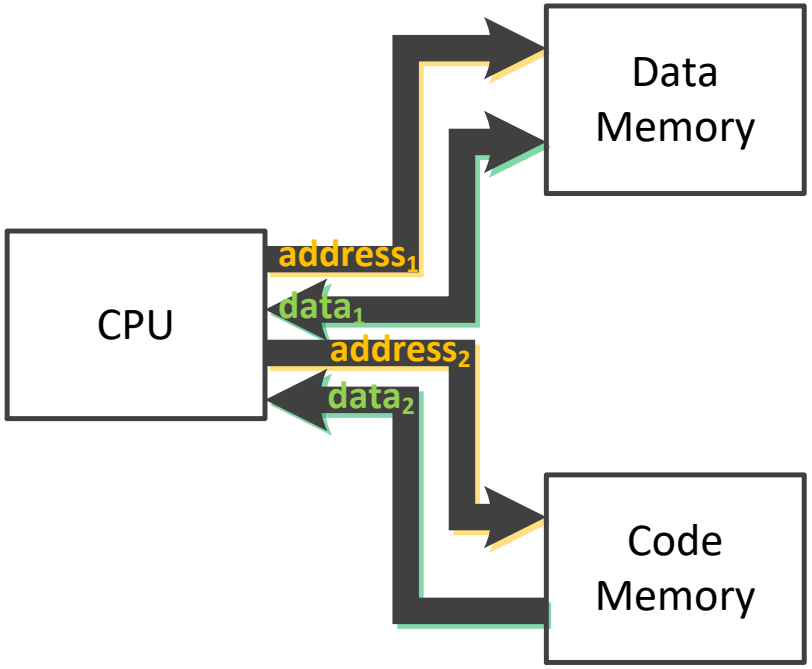
- Aliniere (*alignment*)
  - **stocarea datelor/instrucțiunilor doar la adrese pare**
  - **accesul nealiniat** *poate* **conduce la(depinde de target, compilator):**
    - penalități de timp
    - ocuparea compactă a spațiu de stocare

Memoria – von Neumann vs Harvard

von Neumann



Harvard



Memoria – Clasificare

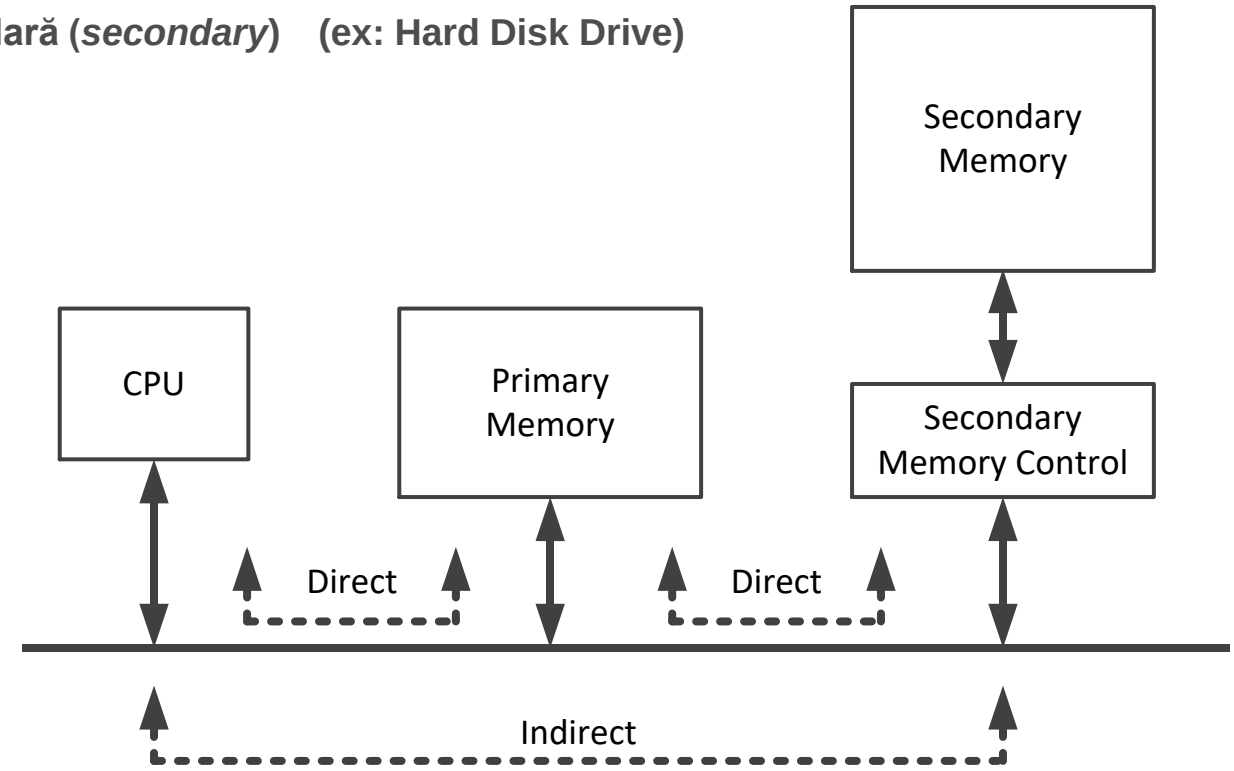
- Tip de acces (din perspectiva CPU):
  - **Read-Write (ex: RAM)**
  - **Read-Only (ex: ROM)**
- Retenția datelor (volatilitate) la scoaterea de sub tensiune:
  - **Volatilă (ex: RAM)**
  - **Nevolatilă (ex: ROM, NVRAM)**
- Accesibilitate
  - ***Sincronă* – citire/scriere doar în anumite cuante de timp (ex: DRAM – între refresh-uri)**
  - ***Asincronă* – citire/scriere oricând (ex: SRAM)**
- Mod de stocare și acces fizic
  - **Secvențial (liniar) (ex: Magnetic Tape)**
  - **Direct (liniar pe mai multe nivele) (ex: Disk)**
  - **Random (matrice) (ex: DRAM, SRAM)**

Memoria – Clasificare

- Acces direct/indirect
  - **memorie principală (în lit. *primary* sau *main*) (ex: RAM)**
  - **memorie secundară (*secondary*) (ex: Hard Disk Drive)**

…booting OS ?

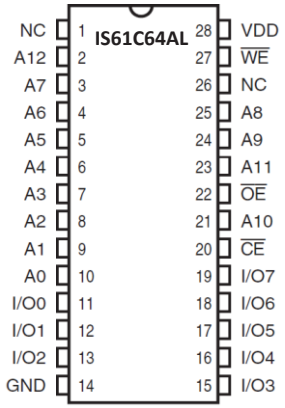
…task execution ?

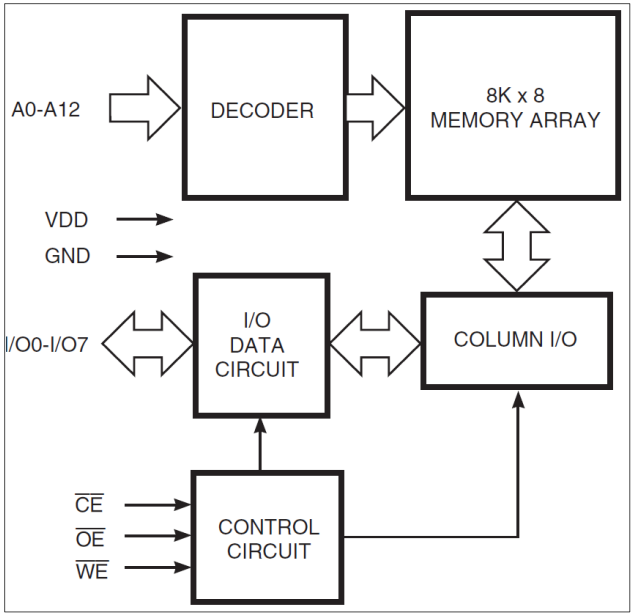


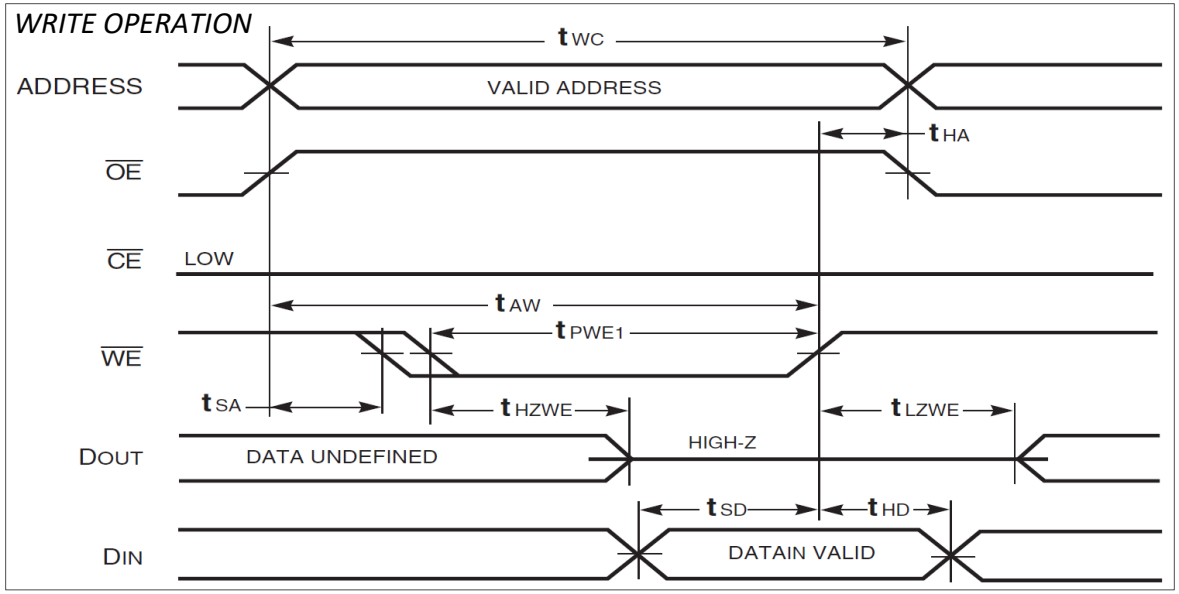
Memoria – Exemple

- Memorie Read-Only (ROM)
  - **mask ROM** – **scriere din fabrică (low-cost)**
  - **PROM (Programmable ROM)** – **scriere cu dispozitiv programator**
  - **EPROM (Erasable PROM)** – **ștergere totală prin expunere la UV**
  - **Flash EPROM** – **ștergere totală electrică**
  - **EEPROM (Electrically Erasable PROM)** – **scriere la nivel de octet**
- Memorie Read-Write
  - **DRAM (Dynamic RAM)**
    - un bit de informație = un condensator + un tranzistor
    - elementul de stocare: condensatorul
      - trebuie reîncărcat atunci când stochează “1” logic, din 2 motive:
        - » pierde sarcină electrică (*charge leakage*)
        - » citirea este un proces destructiv
  - **SRAM (Static RAM)**
    - un bit de informație = un flip-flop = 4 tranzistori
    - elementul de stocare: flip-flop-ul
  - **NVRAM (Non-volatile RAM)**
    - RAM + baterie + circuit de control al bateriei

Memoria – Exemplu – SRAM – IS61C64AL



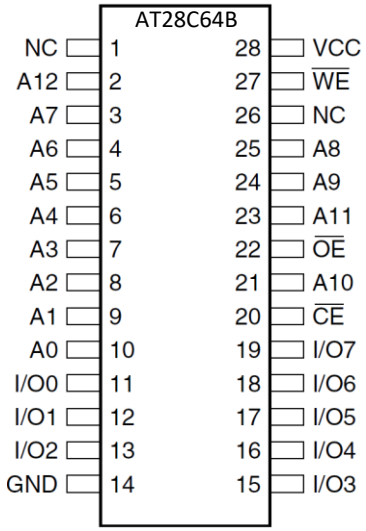


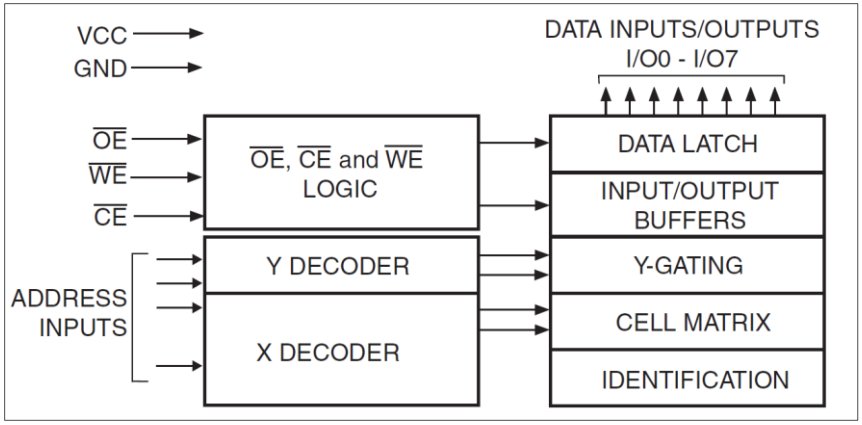


TRUTH TABLE

Mode	$\overline{WE}$	$\overline{CE}$	$\overline{OE}$	I/O Operation	V_{DD} Current
Not Selected (Power-down)	X	H	X	High-Z	I_{SB1}, I_{SB2}
Output Disabled	H	L	H	High-Z	I_{CC}
Read	H	L	L	D_{OUT}	I_{CC}
Write	L	L	X	D_{IN}	I_{CC}

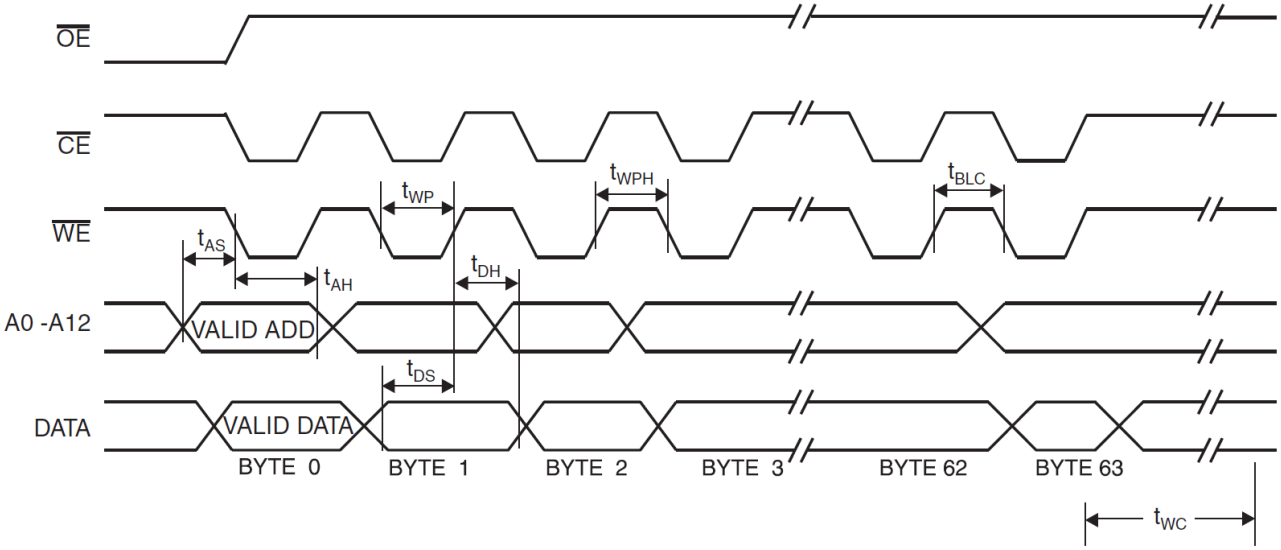
Memoria – Exemplu – EEPROM – AT28C64B





Mode	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	I/O
Read	V_{IL}	V_{IL}	V_{IH}	D_{OUT}
Write[2]	V_{IL}	V_{IH}	V_{IL}	D_{IN}
Standby/Write Inhibit	V_{IH}	X[1]	X	High Z
Write Inhibit	X	X	V_{IH}	
Write Inhibit	X	V_{IL}	X	
Output Disable	X	V_{IH}	X	High Z
Chip Erase	V_{IL}	V_H[3]	V_{IL}	High Z

PAGE WRITE OPERATION



Memoria – Generalități

- Costul unei memorii este invers proporțional cu timpul de acces

- Memoria ideală ?
  - **cost redus**
  - **densitate mare (spațiu de stocare raportat la suprafață)**
  - **RAM**
  - **timp redus de acces**
  - **read/write**
  - **nevolatilă**
  - **număr nelimitat de scrieri/citiri**
  - **imună la erori**

Memoria Cache.
Memoria virtuală

Memoria Cache. Memoria virtuală

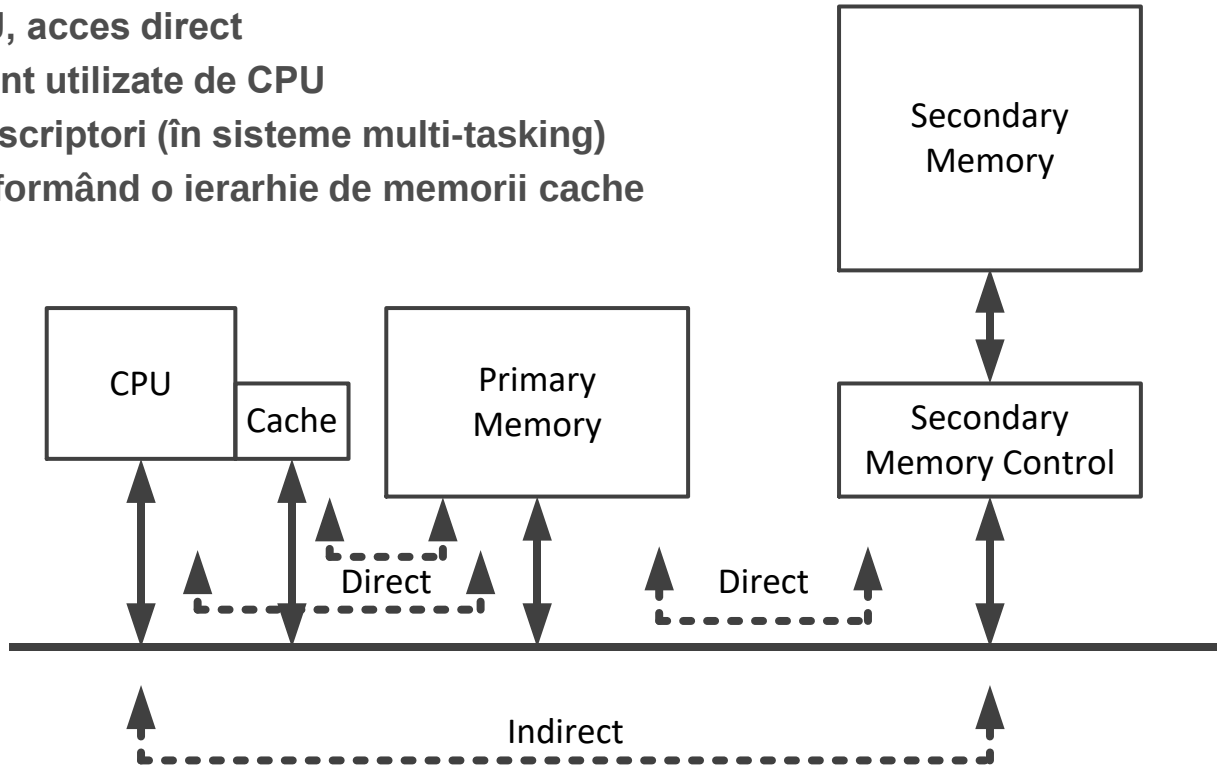
- **Memoria Cache**, respectiv memoria virtuală creează iluzia pentru programator (pentru programul scris de el) că memoria de care dispune este:
  - **mare cât o memorie de tip Disk**
  - **rapidă cât o memorie de tip SRAM**

- Termenul de *memorie Cache* reprezintă o implementare și un concept:
  - **o memorie de tip SRAM**
  - **modul de utilizare al acestei memorii SRAM, în raport cu:**
    - Procesorul
    - Memoria Principală (adică următoarea memorie din ierarhie - SRAM sau DRAM)

- Termenul de *memorie virtuală* reprezintă un concept:
  - **modul de utilizare al Memoriei Principale, în raport cu:**
    - Memoria Cache, Procesorul
    - Memoria Secundară

Memoria – **Cache**

- Caracteristici generale
  - **memorie SRAM cu timp de acces foarte redus (ps, ns)**
  - **capacitate de stocare de ordinul megaocteților**
  - **apropiată fizic de CPU, acces direct**
  - **stochează date frecvent utilizate de CPU**
  - **stochează tabel cu descriptori (în sisteme multi-tasking)**
  - **pot exista mai multe, formând o ierarhie de memorii cache**



Memoria – **Cache**

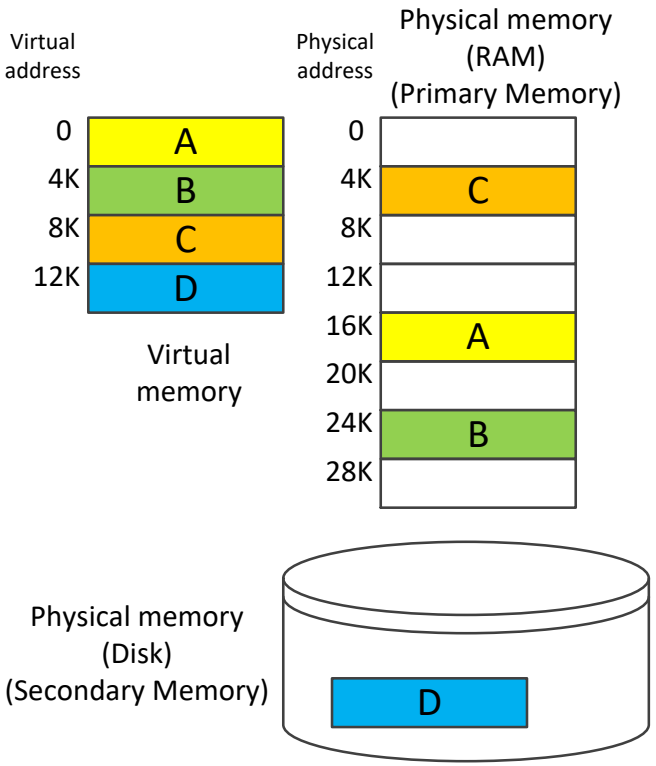
- Terminologie
  - ***block***
  - **nivel în ierarhie (*level in memory hierarchy)***
  - ***locality (spatial, temporal)***
  - ***hit*, *miss***
  - ***hit rate*, *miss rate* (1 – *hit rate*)**

- Moduri de asociere (mapare) a datelor între nivele consecutive ale ierarhiei
  - **mapare directă**
  - **mapare asociativă**
  - **mapare set-asociativă (*one-way*)**
  - **mapare set-asociativă (*two-way*)**

Memoria virtuală

- Memoria virtuală
  - **procesul "vede" memoria alocată pentru el (în RAM, de către sistemul de operare) ca pe un spațiu continuu de adrese**
  - **de fapt, sistemul de operare poate întârzia stocarea unor pagini în RAM până când acele pagini vor fi necesare procesului**

- Terminologie
  - **adresă fizica, adresă virtuală**
  - ***address translation***
  - ***page, page fault***
  - ***protection***
    - Un alt task aflat în execuție va putea accesa doar codul și datele proprii
  - ***segmentation***
  - ***swap space***

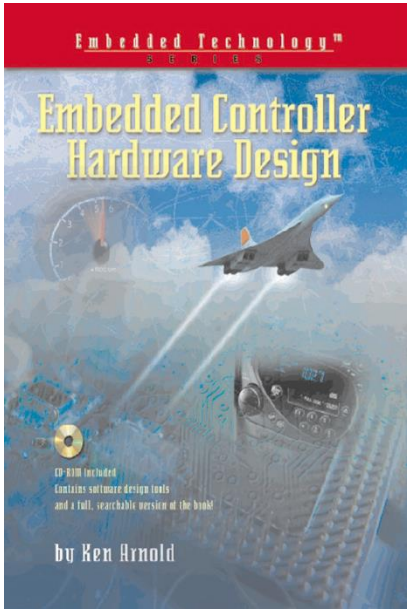


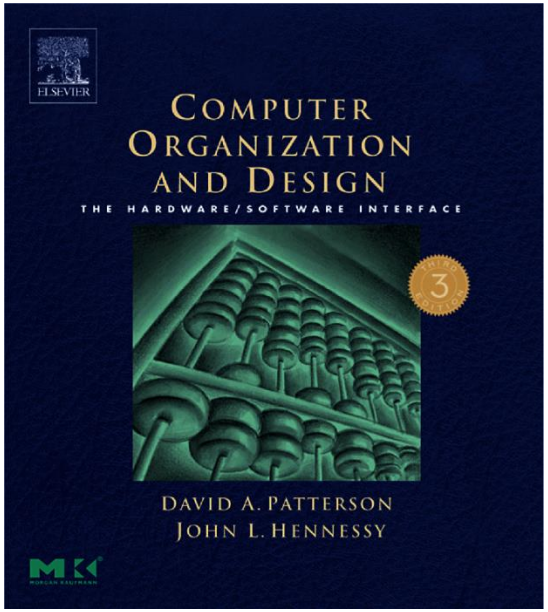
Memoria – Erori

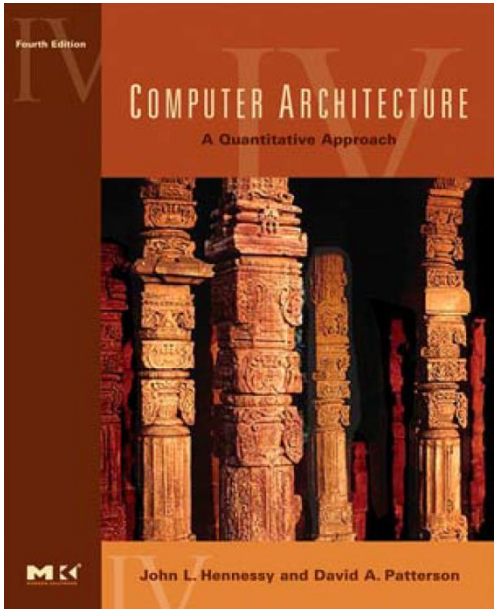
- O clasificare a erorii:
  - **Hard error: se întâmplă mereu (*periodic*)**
  - **Soft error: se întâmplă sporadic (*transient*)**
- Detecția erorii
  - **Metode implementate fie în Hardware, fie în Software, fie în combinație**
  - **efect: oprește operația de citire înainte ca acea dată să fie folosită**
  - **bit de paritate**
  - **sumă de control**
  - **CRC**
- Corecția erorii
  - **Metode implementate fie în Hardware, fie în Software, fie în combinație**
  - **efect: utilizează informații redundante pentru refacerea datei afectate de eroare**
  - **bloc de paritate**
  - **ECC (coduri corectoare de erori)**

Memoria – Generalități – Referințe

- *Embedded Controller Hardware Design*, Ken Arnold, 2000
- Hennesy & Patterson:
  - ***Computer Organization and Design – The Hardware Software Interface*, 3rd ed, 2004**
  - ***Computer Architecture – A Quantitative Approach*, 4th ed, 2007**





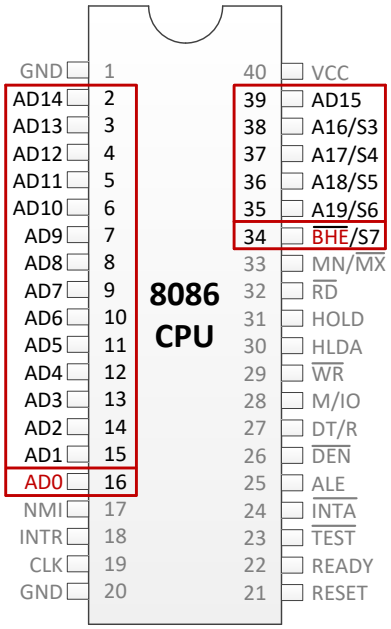


Intel 8086

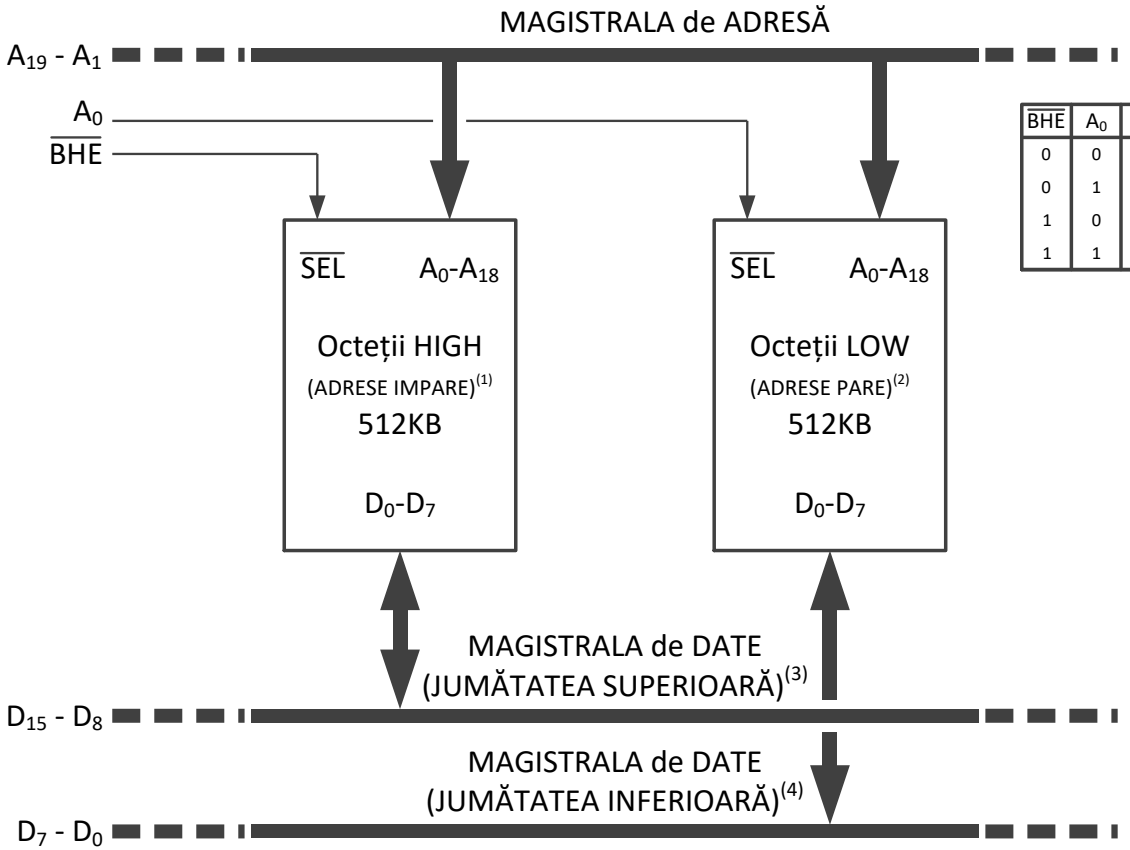
CPU

8086 – Interfațarea cu memoria (externă)

- AD0 - AD15 pini de intrare/ieșire
  - **citire/scriere date**
  - **citire instrucțiuni**
  - **parte din adresa fizică**
- AD16 - AD19 pini de ieșire
  - **parte din adresa fizică**
- 16+4 biți → 2^{20} = 1MB spațiu adresabil
- $\overline{\text{BHE}}$ - Bus High Enable
  - **Împreună cu AD0 realizează selecția memoriei** (slide următor)

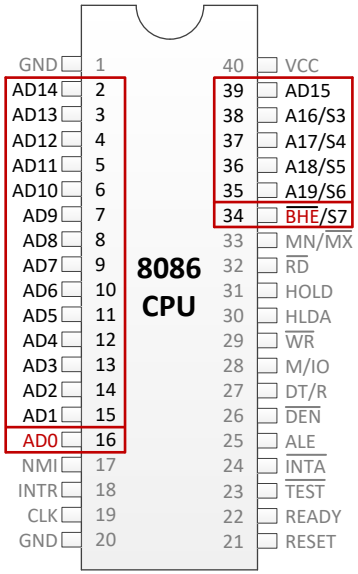


8086 – Interfațarea cu memoria (externă)



$\overline{BHE}$	A_0	TRANSFER
0	0	ambii octeți
0	1	octetul superior (de la adresa impară)
1	0	octetul inferior (de la adresa pară)
1	1	nimic

(1) UPPER (ODD) BANK
(2) LOWER (EVEN) BANK
(3) UPPER HALF OF DATA BUS
(4) LOWER HALF OF DATA BUS

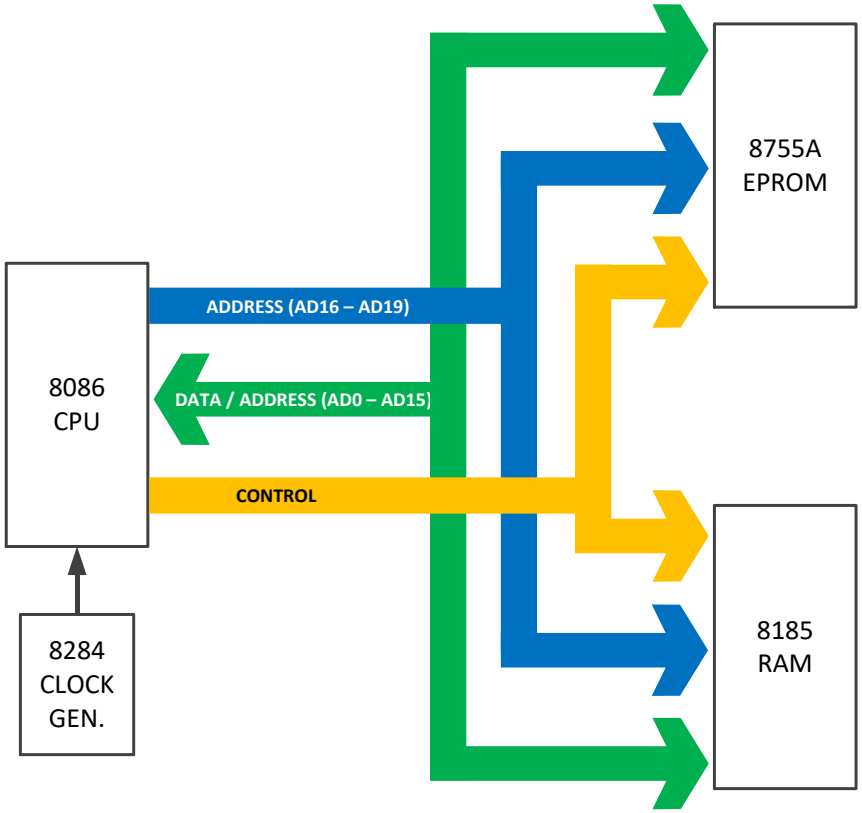


8086 – Interfațarea cu memoria (externă)

- Arhitectură de tip **von Neumann**, deoarece:
  - **există o singură magistrală prin care CPU poate citi/scrie date, respectiv extrage instrucțiuni**

  (memoriile de cod, date sunt conectate la aceeași magistrală de date/adrese)

- Spațiul de memorie este unul continuu la nivel de adresă fizică

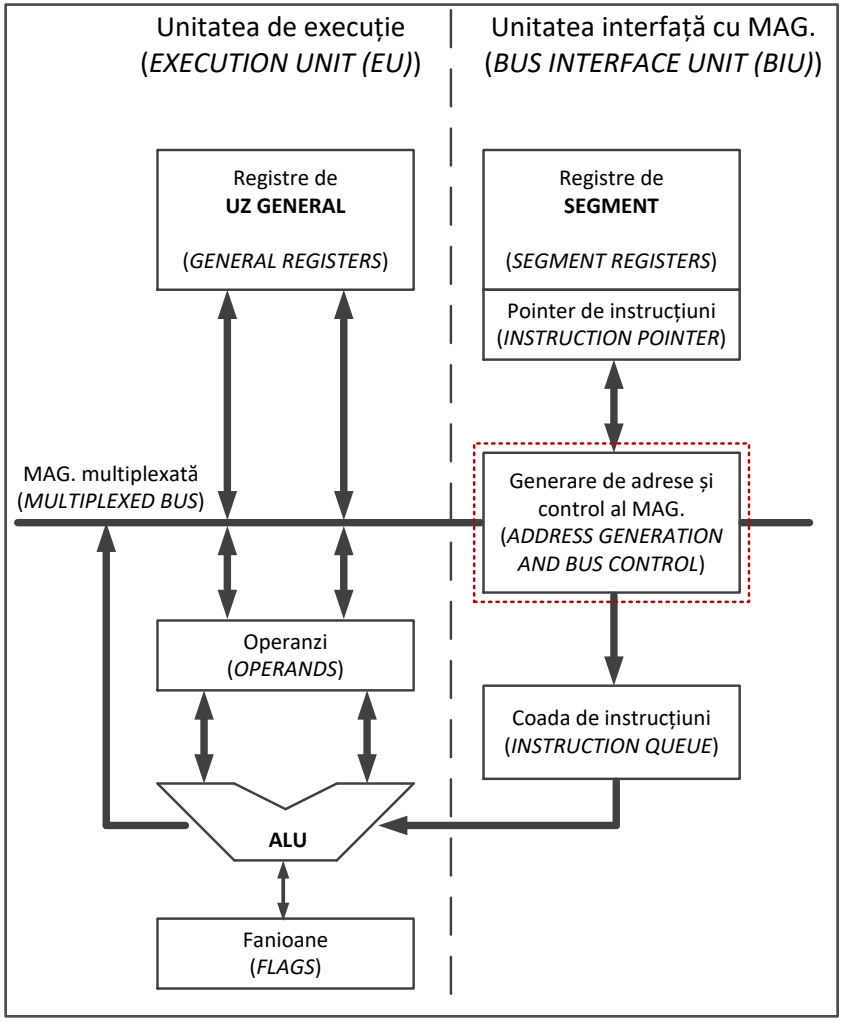


8086 – *Execution Unit*, *Bus Interface Unit*

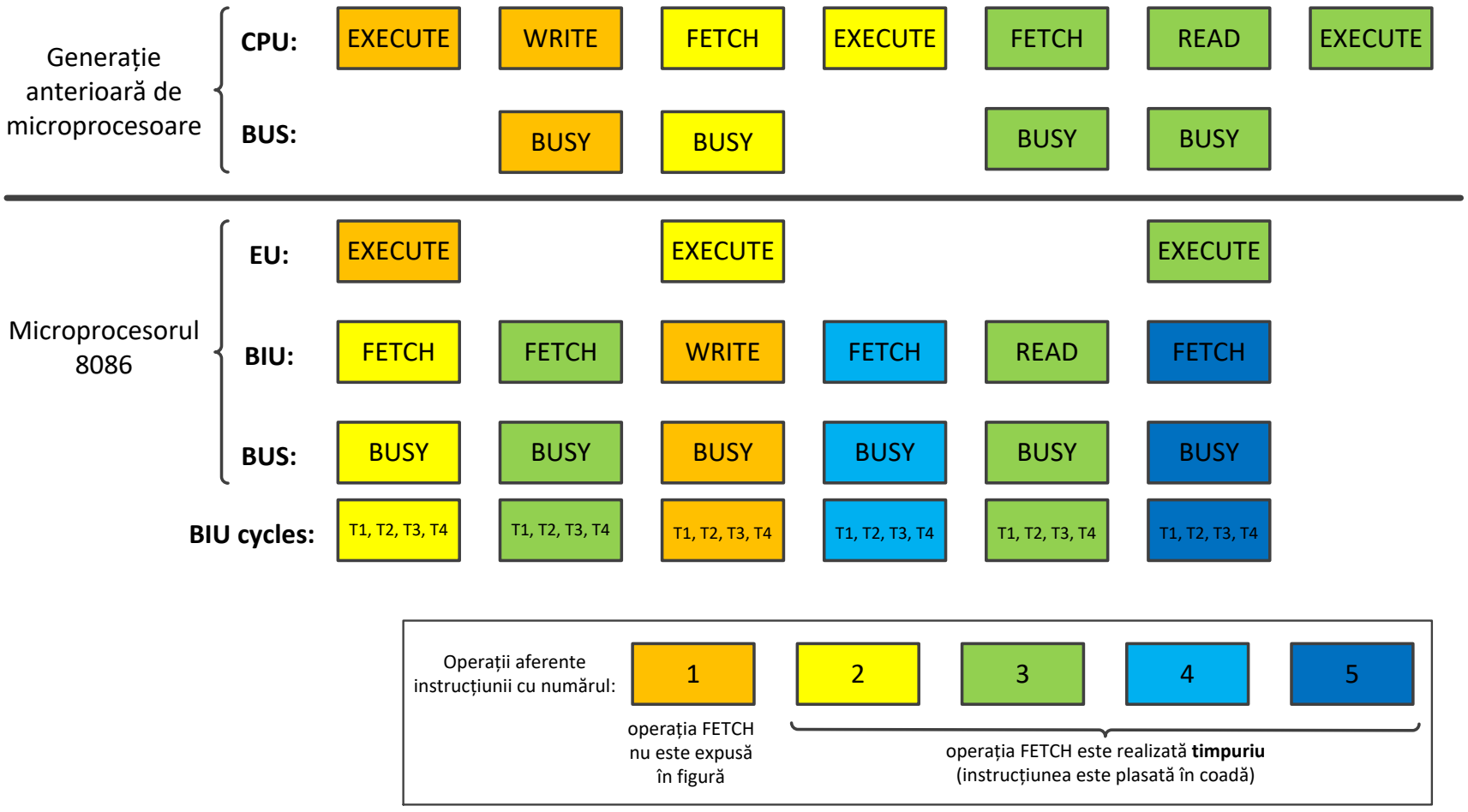
- AGBC comunică cu memoria

- *IP* îi furnizează adresa instrucțiunii
- AGBC preia instrucțiunea din memorie
- *EU* preia instrucțiunea din coadă

- Dacă instrucțiunea necesită acces la memorie: *EU* cere AGBC acele date

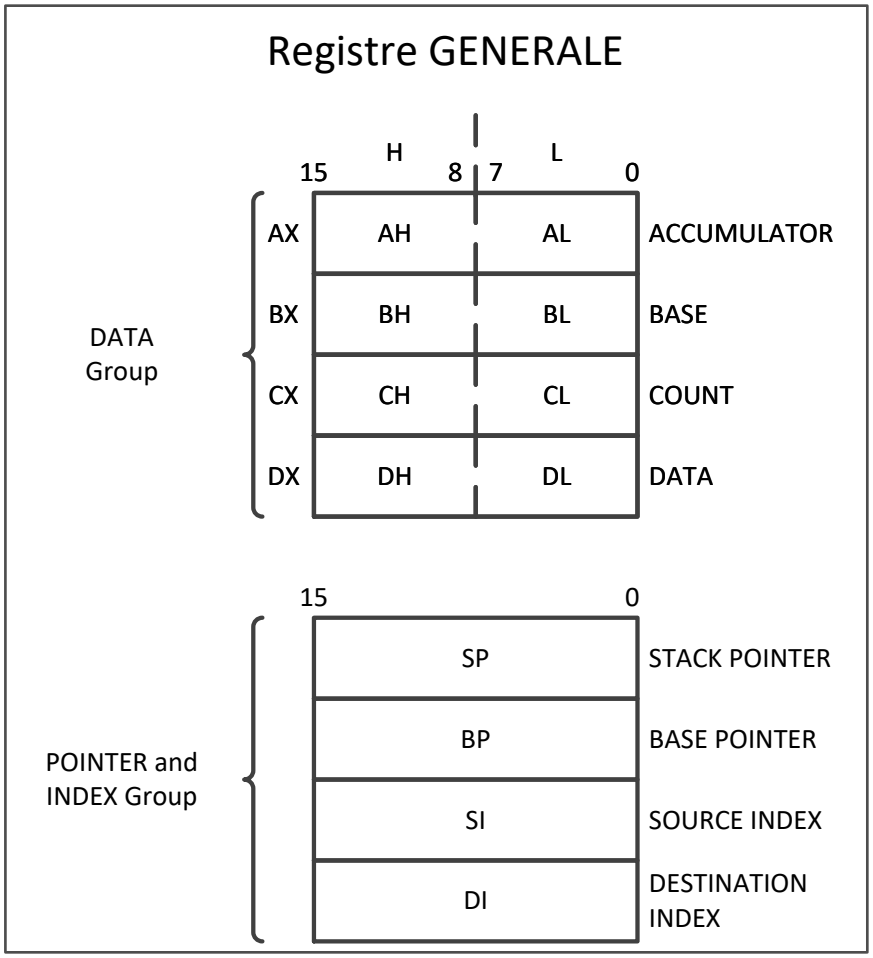


8086 – Funcționarea suprapusă a *EU* și *BIU*



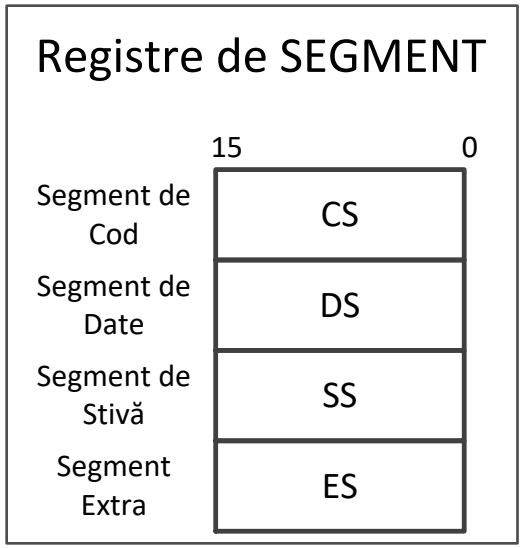
8086 – Registrele generale ale CPU

- Registre generale
  - **AX, BX, CX, DX**
    accesibile pe 8 bit: H și L
  - **utilizate implicit în anumite instrucțiuni**



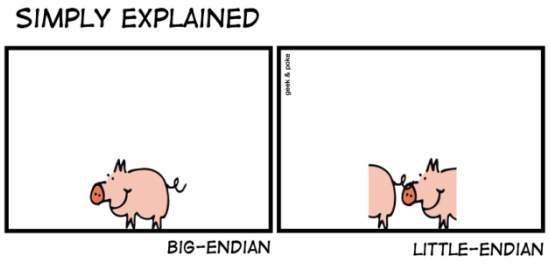
8086 – Registrele de segment ale CPU

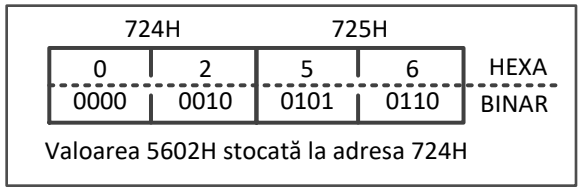
- Registre de segment
  - **accesibile doar pe 16 biți**
- Pointerul de instrucțiuni (*IP*)
  - **stochează deplasamentul(*offset*) față de valoarea din CS**
    - CS:IP = adresa următoarei instrucțiuni
- Memoria este împărțită la nivel logic în segmente de 2^{16} octeți = 64 kB
- 4 registre de segment
  - **4 segmente logice (date, cod) accesibile la un moment dat de către CPU**

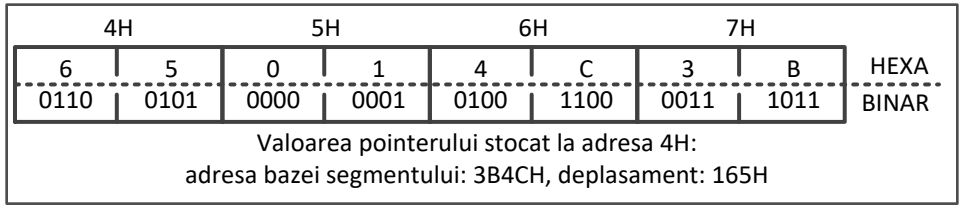


8086 – Organizarea memoriei

- Little-endian
  - **exemplificare pentru cuvânt (2 octeți)**
  - **exemplificare pentru cuvânt dublu (4 octeți)**
    - Pointer - folosit pentru adresarea de date/instrucțiuni în afara segmentului curent



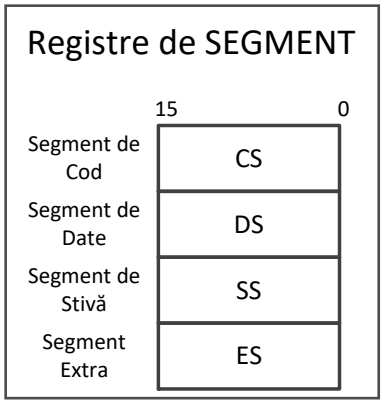


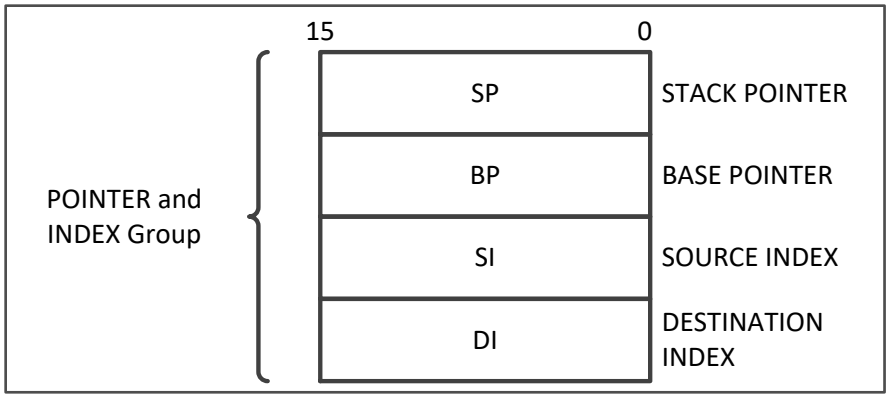


- Nu există constrângeri de aliniament
  - **beneficiu: utilizarea întregului spațiu, fără octeți nefolosiți (irosiți)**
  - **cuvântul (2 octeți) stocat nealiniat necesită 2 operații de transfer (vezi slide-ul cu interfața cu memoria)**

8086 – Organizarea memoriei

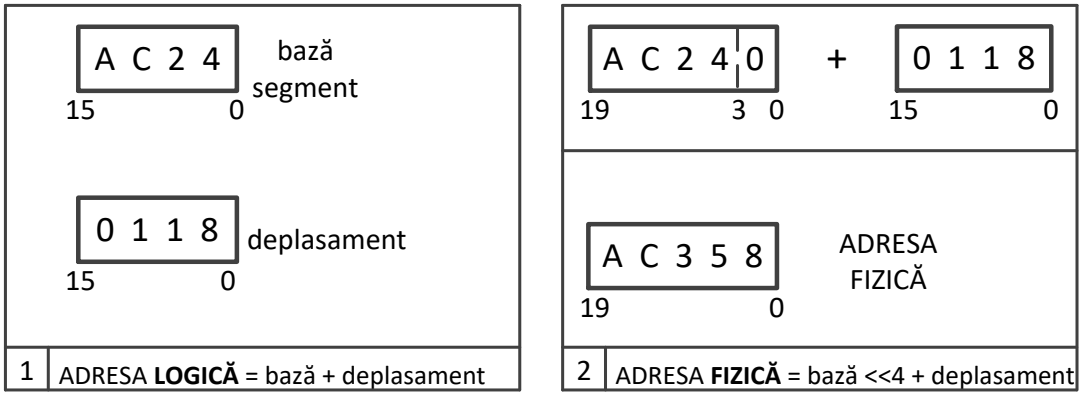
- Memoria este împărțită **la nivel logic** în segmente
  - **baza/începutul segmentului = valoarea/adresa din registrul de segment**
  - **inițializarea bazei segmentului se face în software funcție de modelul de memorie utilizat**
  - **un segment poate avea maxim 64kB**
  - **segmentele se pot suprapune parțial sau total**
  - **accesul în segmentul curent se face mai rapid decât accesul în afara lui**
- Adresa logică = bază segment (16 biți) + deplasament (16 biți)

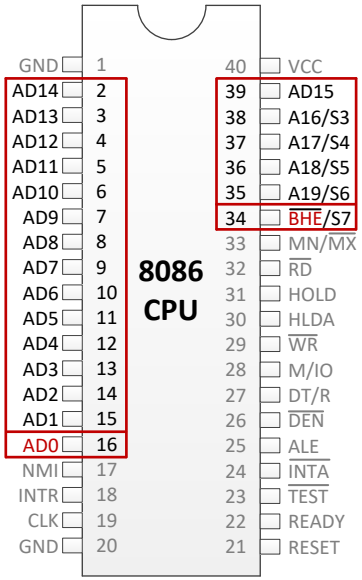




8086 – Formarea adresei fizice

- Adresa logică = bază segment (16 biți) + deplasament (16 biți)
- Adresarea fizică a memoriei se realizează prin cei 21 de pini fizici
- Datorită modului de calcul al adresei fizice,
  orice segment începe la o adresă multiplu de 16





8086 – Moduri de adresare

- Mod de adresare
  - **modul în care se calculează adresa fizică a operandului**
- Instrucțiunile care au ca operanzi registre sau constante:
  - **citirea/scrierea în registru se face direct de către ALU**
  - **constantele sunt preluate din codificarea instrucțiunii**
  - **cele mai rapide instrucțiuni pentru că nu necesită comunicație pe magistrală**
  - **cele mai compacte instrucțiuni deoarece registrele au o adresare pe 8 biți**

- Moduri de adresare:
  - **imediată**
  - **directă**
  - **indirectă prin registru**
  - **bazată**
  - **indexată**
  - **bazată și indexată**

8086 – Adresare imediată. Adresare directă

- Adresare imediată
  - **operandul apare explicit în codificarea instrucțiunii (operandul este o constantă, o valoare imediată)**

```
mov     AX, 1
```

- Adresare directă
  - **operandul este un *offset* față de adresa indicată de registrul de segment (implicit DS)**

```
.data
        val     dw      1
.code
        mov     BX, val     ; la asamblare, val va fi înlocuit cu offset-ul în segmentul de date

        add     BX, [100]   ; operandul se află în segmentul de date la offset-ul 100
```

8086 – Adresare indirectă. Adresare bazată

- Adresare indirectă
  - **adresa operandului este suma dintre conținutul unui registru (BX, SI sau DI) și conținutul unui registru de segment (implicit DS)**

```
mov     AX, [BX]            ; pune in AX conținutul locației indicate de adresa din BX
mov     AX, SS:[SP]         ; pune în AX vârful stivei (fără a altera stiva, adică SP)
```

- Adresare bazată(1) sau indexată(2)
  - **adresa operandului este suma dintre conținutul unui registru de bază / index și o adresă de segment**

```
.data
        db      abc 5 dup 0     ; declară un vector de 5 octeți inițializați cu 0
.code
        mov BX, 2
        mov DI, 1
        mov AX, abc[BX]         ; (1) pune în AX octetul de pe poziția 2 din vectorul abc
        mov AX, abc[DI]         ; (2) pune în AX octetul de pe poziția 1 din vectorul abc
```

8086 – Adresare bazată și indexată

- Adresare bazată și indexată
  - **adresa operandului este suma dintre trei termeni: conținutul unui registru de bază, conținutul unui registru de index, și un deplasament valoare imediata**

```
mov AX, [BX][DI][4]
; echivalent cu:

mov AX, [BX+SI+4]
; echivalent cu:

mov AX, DS:[BX+SI+4]
; echivalent cu:

mov AX, [BX+SI].4
```

8086 CPU – Referințe

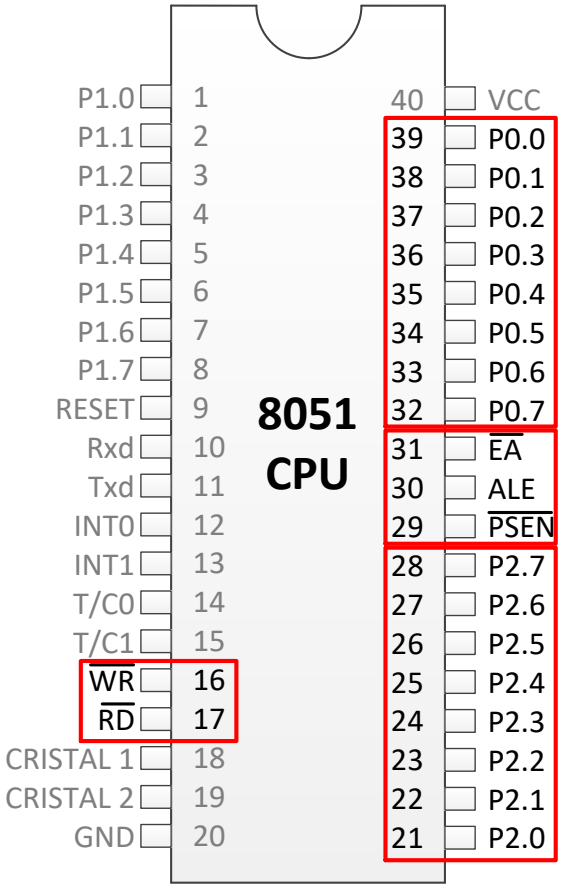
- The 8086 Family User's Manual, Intel Corporation, October 1979
- The Intel Microprocessors 8th Edition, B.B. Brey, Pearson Education, 2009
- https://usermanual.wiki/Document/Intel8086FamilyUsersManual.2660397397/view
- https://userpages.umbc.edu/~squire/intel_book.pdf
- https://en.wikipedia.org/wiki/Intel_8086

Intel 8051 Core

8051 – Memoria

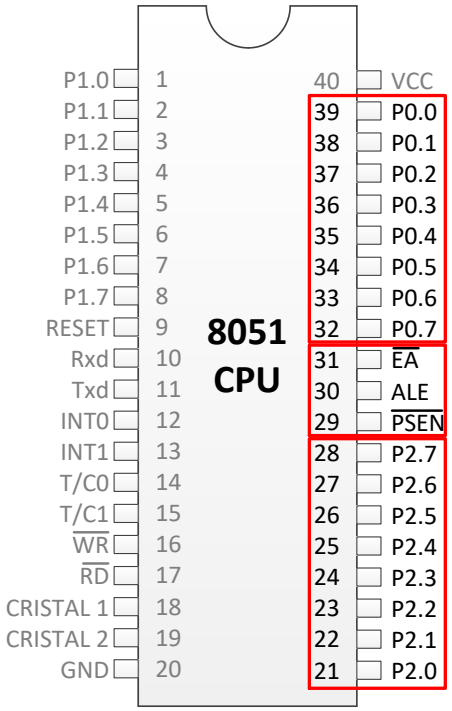
- Accesul la memoria de date prin:
  - **pinii de control RD și WR (*Read* respectiv *Write*)**
  - **adresă 8-bit**
    (instrucțiunea are ca operand un registru)
  - **adresă 16-bit**
    (instrucțiunea are ca operand o locație din memorie)
- Accesul la memoria de program prin:
  - **semnalul de control PSEN (*Program Store Enable*)**
  - **semnalul EA (*External Address*)**
  - **adresă 16-bit**

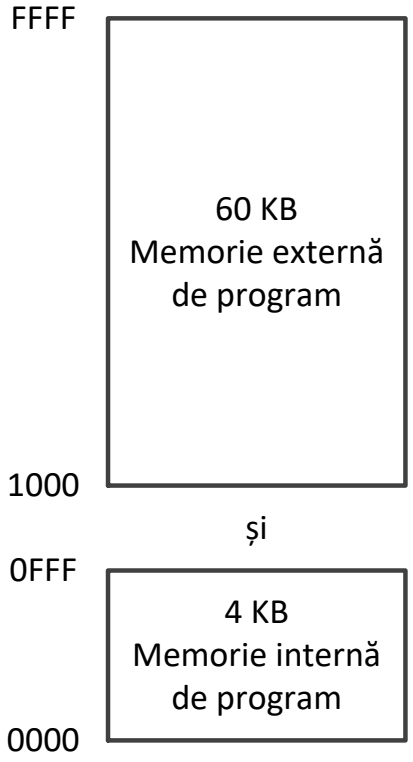
- Adresa 8-bit: pinii portului P0
- Adresa 16-bit este este compusă din:
  - **octetul high: pinii portului P2 (P2.0 – P2.7)**
  - **octetul low: pinii portului P0 (P0.0 – P0.7)**

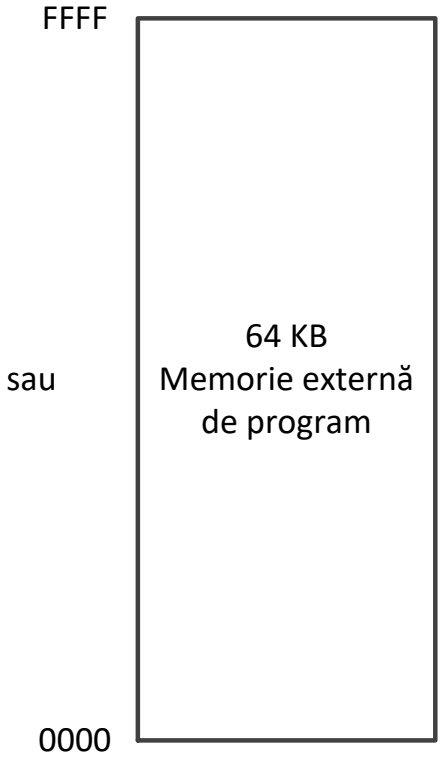


8051 – Memoria – Organizare

- Există o variantă a circuitului 8051 fără memorie de program integrată
- Spațiul de memorie de program
  - **2 posibile configurații**

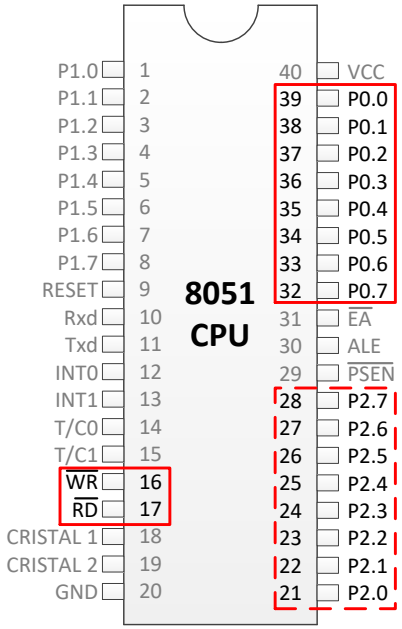


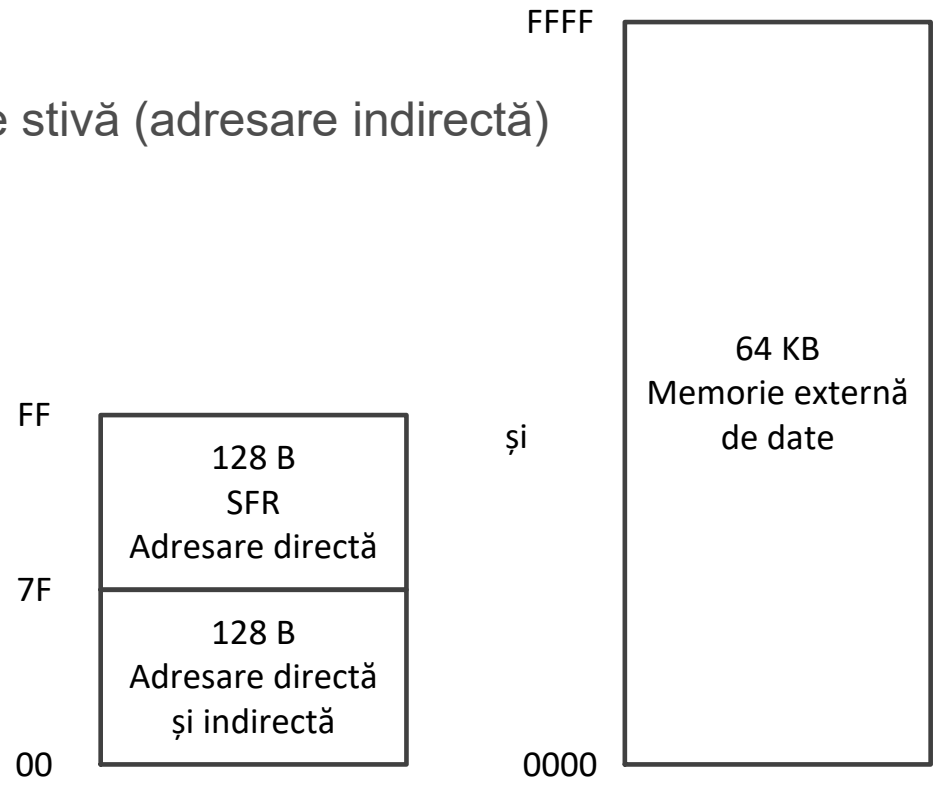




8051 – Memoria – Organizare

- Spațiul extern se accesează printr-o instrucțiune dedicată – MOVX
- Spațiul de memorie de date
  - **memorie de 256 B integrată**
  - **până la 64 KB externi**
- Primii 128 B pot fi utilizați pe post de stivă (adresare indirectă)





8051 – Memoria – Tipuri de adresare

- Adresare directă. Adresa indirectă … *pointer?*

MOV 80H, #0AAH ; scrie 0x0AA in SFR-ul de la adresa 0x80

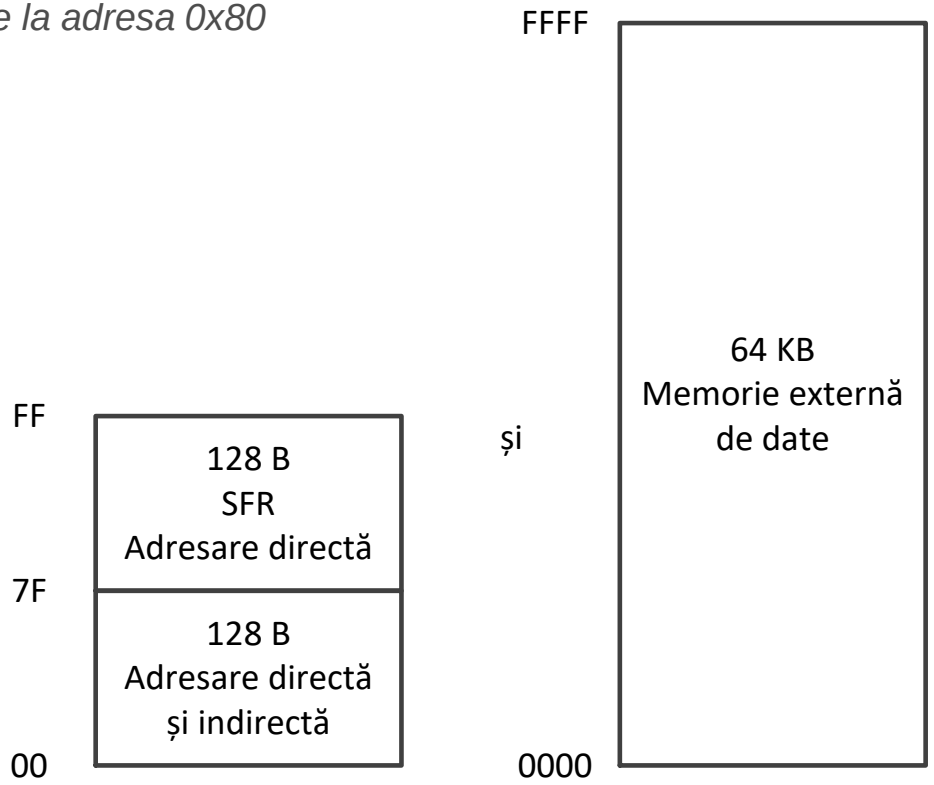
MOV R0, #80H

MOVX @R0, #0BBH ; adresare indirectă

; scrie 0x0BB la adresa 0x80 din memoria externă

; rezultat identic cu:

MOVX 80H, #0BBH ; adresare directă



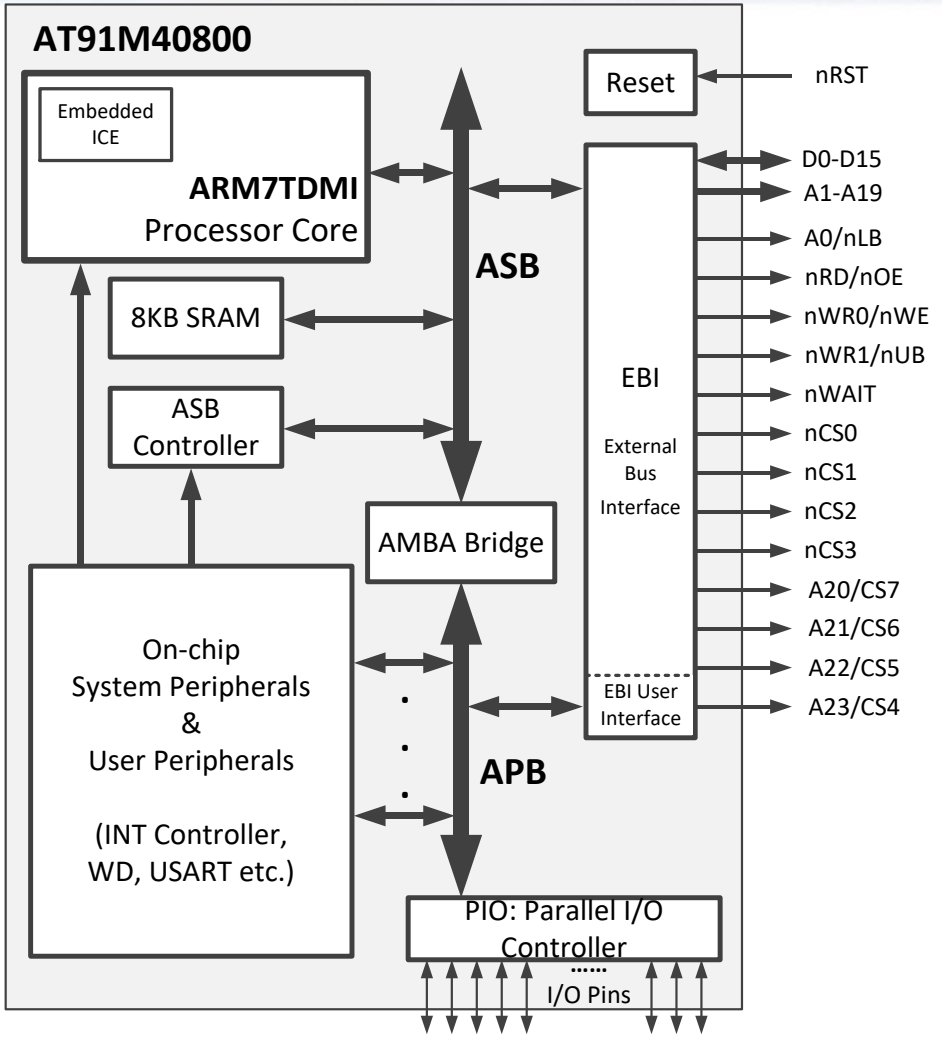
8051 Core – Referințe

- Intel MCS51 Microcontroller Family User's Manual, 1994
- http://web.mit.edu/6.115/www/document/8051.pdf
- https://en.wikipedia.org/wiki/Intel_MCS-51

(Classic) ARMv4 Architecture

ARM7TDMI Core în **AT91M40800** MCU

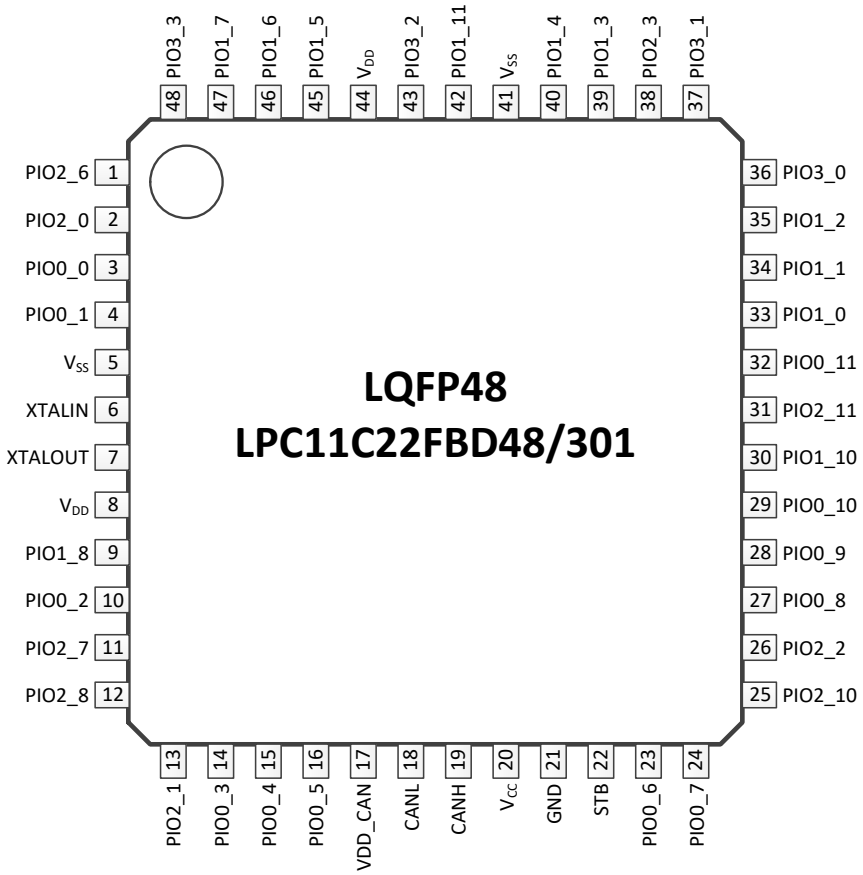
- **AT91M40800**
  - **MCU fără memorie Flash încorporată**
- EBI – External Bus Interface
  - **interfață de magistrală ce permite conectarea cu diverse alte module (memorii și/sau alte periferice)**
  - **D0 - D15 magistrală de date**
  - **A0 - A19 magistrală de adrese**
- ASB – Advanced System Bus
  - **magistrala sistem (internă)**
- APB – Advanced Peripheral Bus
  - **magistrala perifericelor (internă)**
- AMBA™ Bridge
  - **puntea de legătură între ASB și APB**
  - **controlează APB și gestionează cererile de scriere/citire asupra perifericelor**



ARMv6-M Architecture

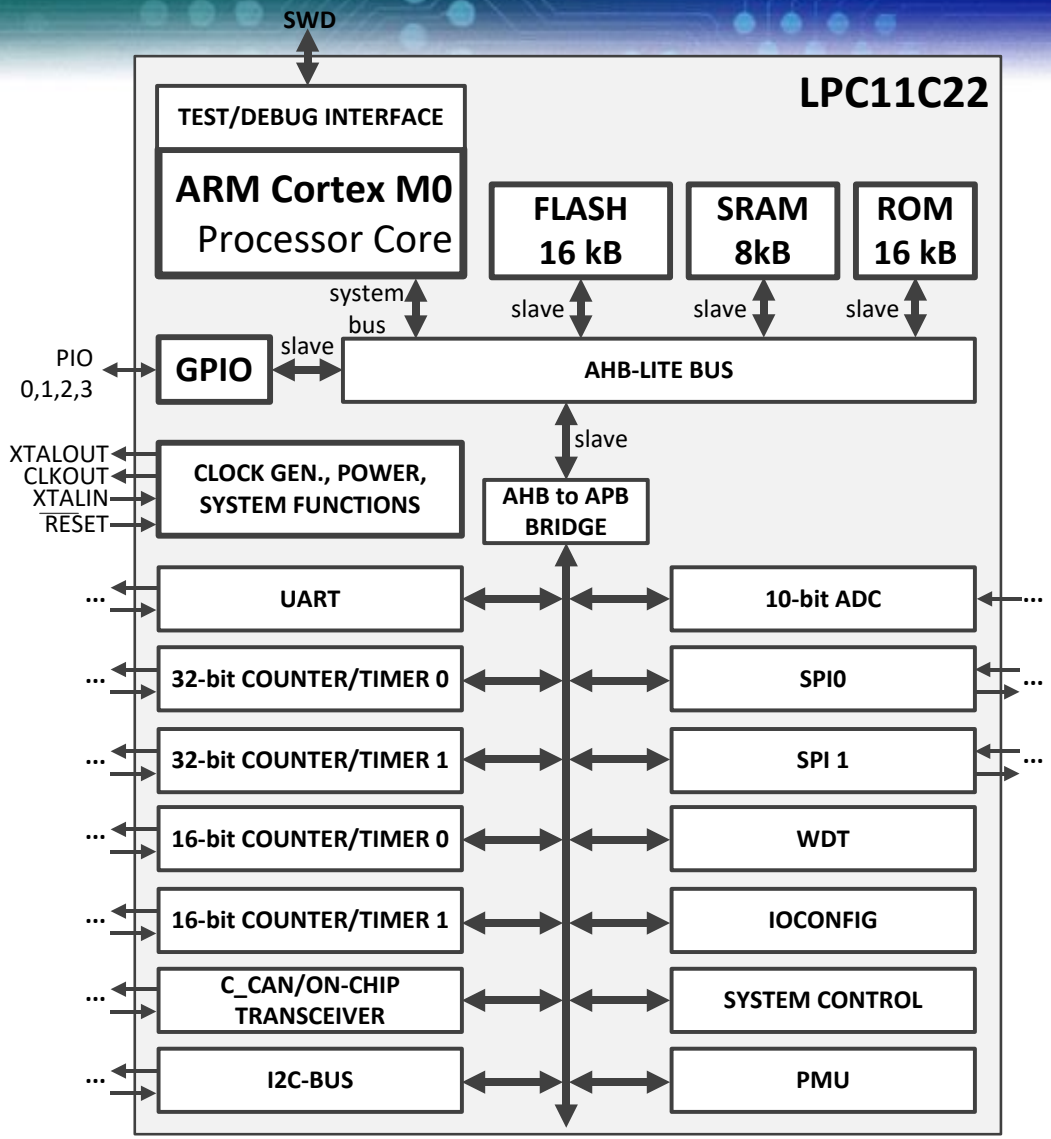
ARM Cortex M0 Core în **LPC11C22** MCU

- **Una dintre implementările arhitecturii ARMv6-M: ARM Cortex M0**
  - **Processor Core integrat în MCU**
    - **exemplu de MCU: LPC11C22 de la NXP**
- **Configurația pinilor MCU**
  - **4 porturi: PIO 0,1,2,3**



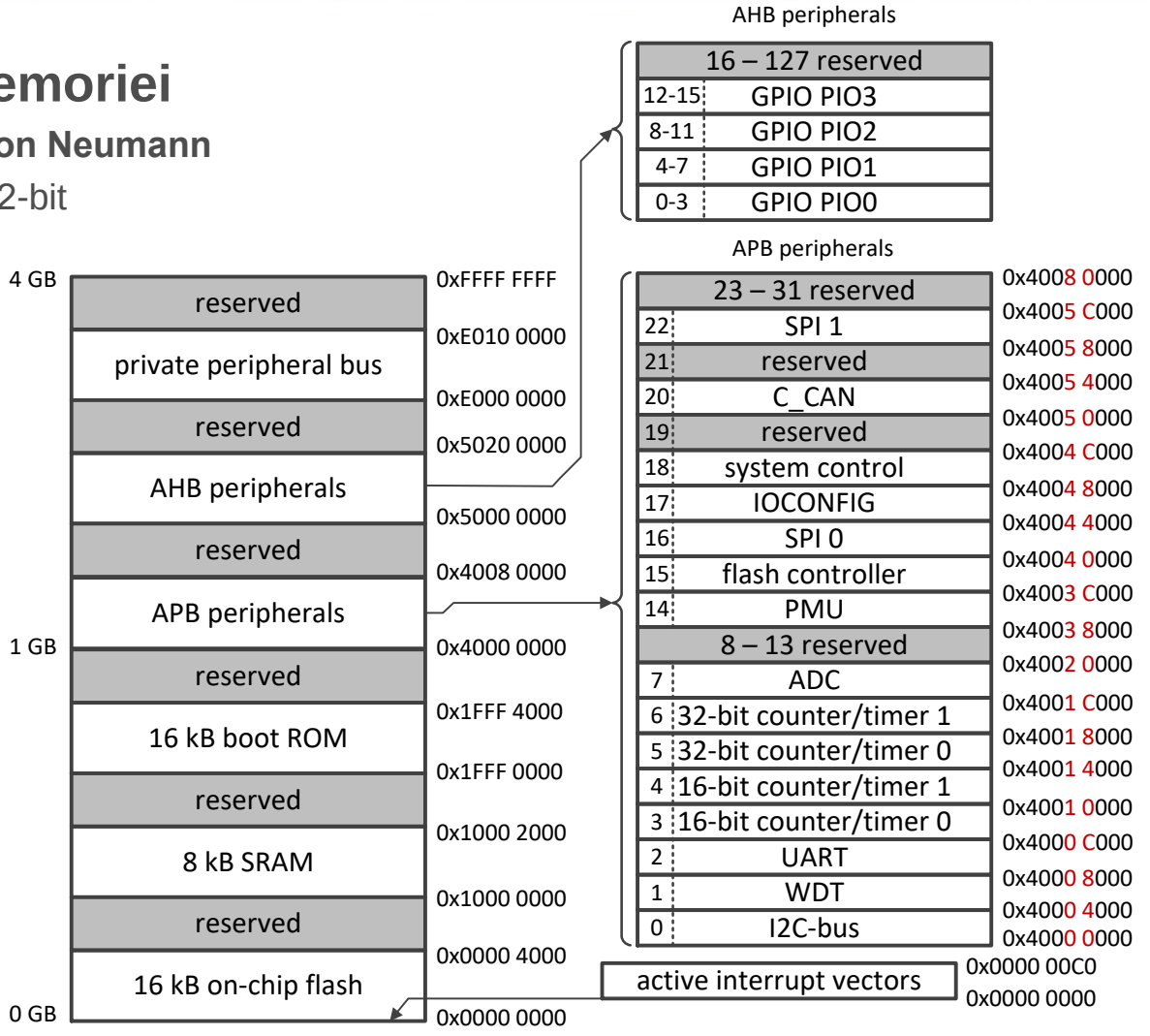
ARM Cortex M0 Core în **LPC11C22** MCU

- **LPC11C22**
  - **Diagrama bloc**



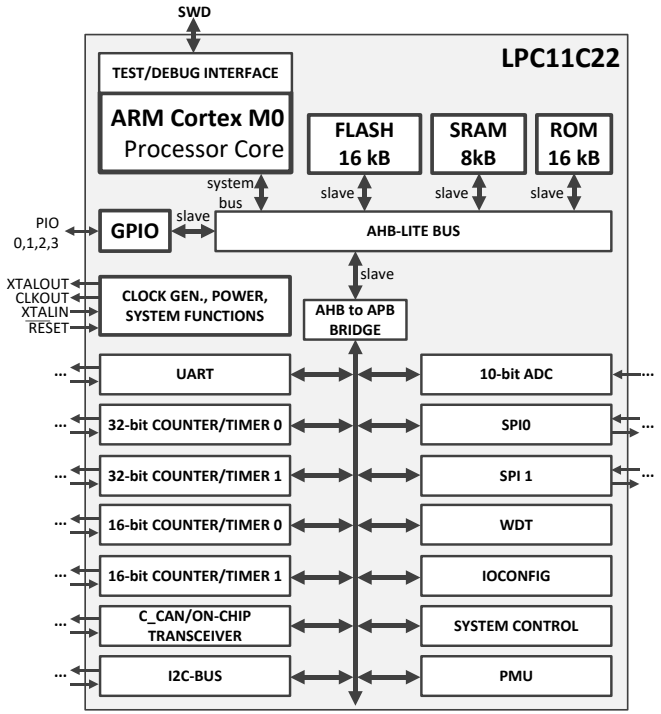
ARM Cortex M0 Core în **LPC11C22** MCU

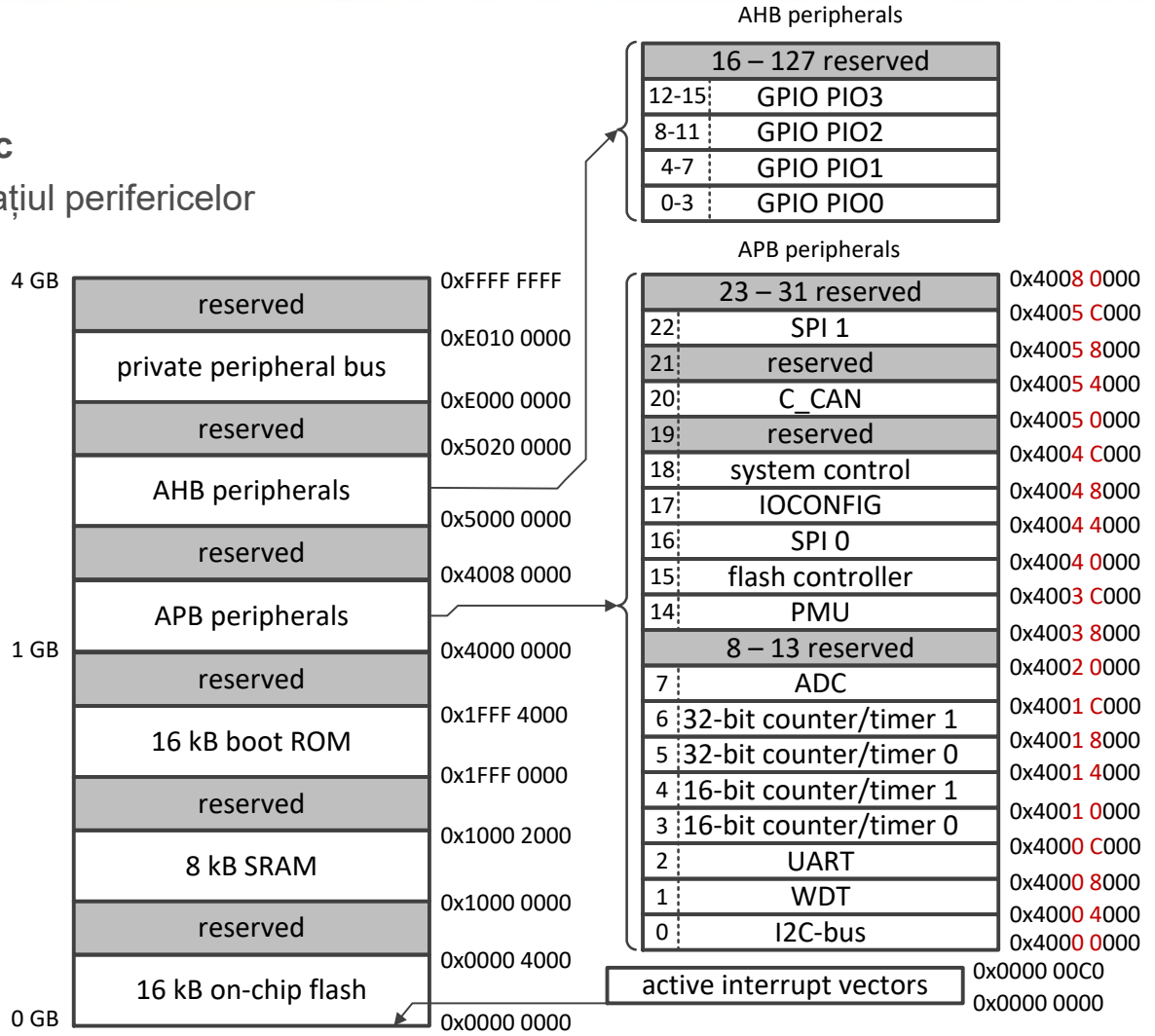
- **LPC11C22 – Harta memoriei**
  - **Cortex M0 - Arhitectură von Neumann**
    - 4GB ↔ adresare pe 32-bit



ARM Cortex M0 Core în **LPC11C22** MCU

- **Harta memoriei**
  - **legătura cu diagrama bloc**
    - spațiul memoriilor, spațiul perifericelor





ARM Cortex-M Architectures

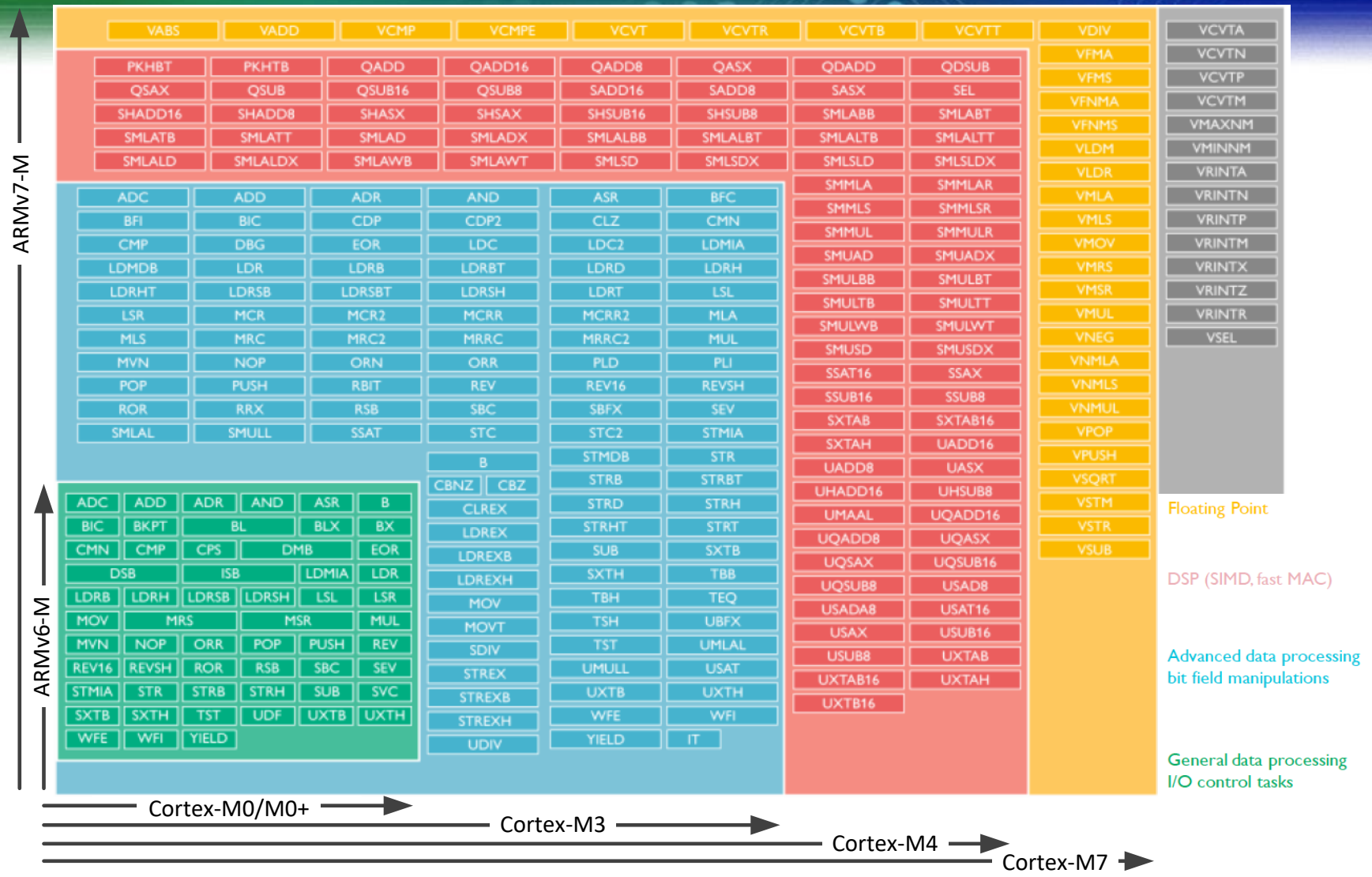
ARM Cortex M

Feature \ Core	M0	M0+	M1	M23	M3	M4	M33	M35P	M55	M7
ISA (ARMv...-M)	6	6	6	8, Baseline	7	7E	8, Mainline	8, Mainline	8, Mainline	7E
TrustZone for Armv8-M	-	-	-	Opt.	-	-	Opt.	Opt.	Opt.	-
DSP Extension	-	-	-	-	-	Yes	Yes	Yes	Yes	Yes
Hardware Divide	-	-	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ARM custom instructions	-	-	-	-	-	-	Yes	-	Yes	-
Coprocessor Interface	-	-	-	-	-	-	Yes	Yes	Yes	-
DMIPS / MHz	0.87	0.95	0.8	0.98	1.25	1.25	1.5	1.5	1.6	2.14
CoreMark / MHz	2.33	2.46	1.85	2.64	3.34	3.42	4.02	4.02	4.2	5.01
Max. External Interrupts	32	32	32	240	240	240	480	480	480	240
Max. MPU Regions	0	8	0	16	8	8	16	16	16	16
Bus Protocol	AHB Lite	AHB Lite	AHB Lite	AHB5	AHB Lite	AHB Lite	AHB	AHB	AXI	AXI
Instruction Cache (kB)	-	-	-	-	-	-	-	2-16	0-64	0-64
Data Cache	-	-	-	-	-	-	-	-	0-64	0-64
Instruction TCM (MB)	-	-	-	-	-	-	-	-	0-16	0-16
Data TCM (MB)	-	-	-	-	-	-	-	-	0-16	0-16
Dual Core Lock-Step (DCLS)	-	-	-	Yes	-	-	Yes	Yes	-	Yes
Common Criteria Certification	-	-	-	-	-	-	Yes	Yes	-	-
Reference Package and/or System Example	Corstone -101	Corstone -101	-	Corstone -102	Corstone -101	Corstone -101	Corstone -201	-	Corstone -300	-

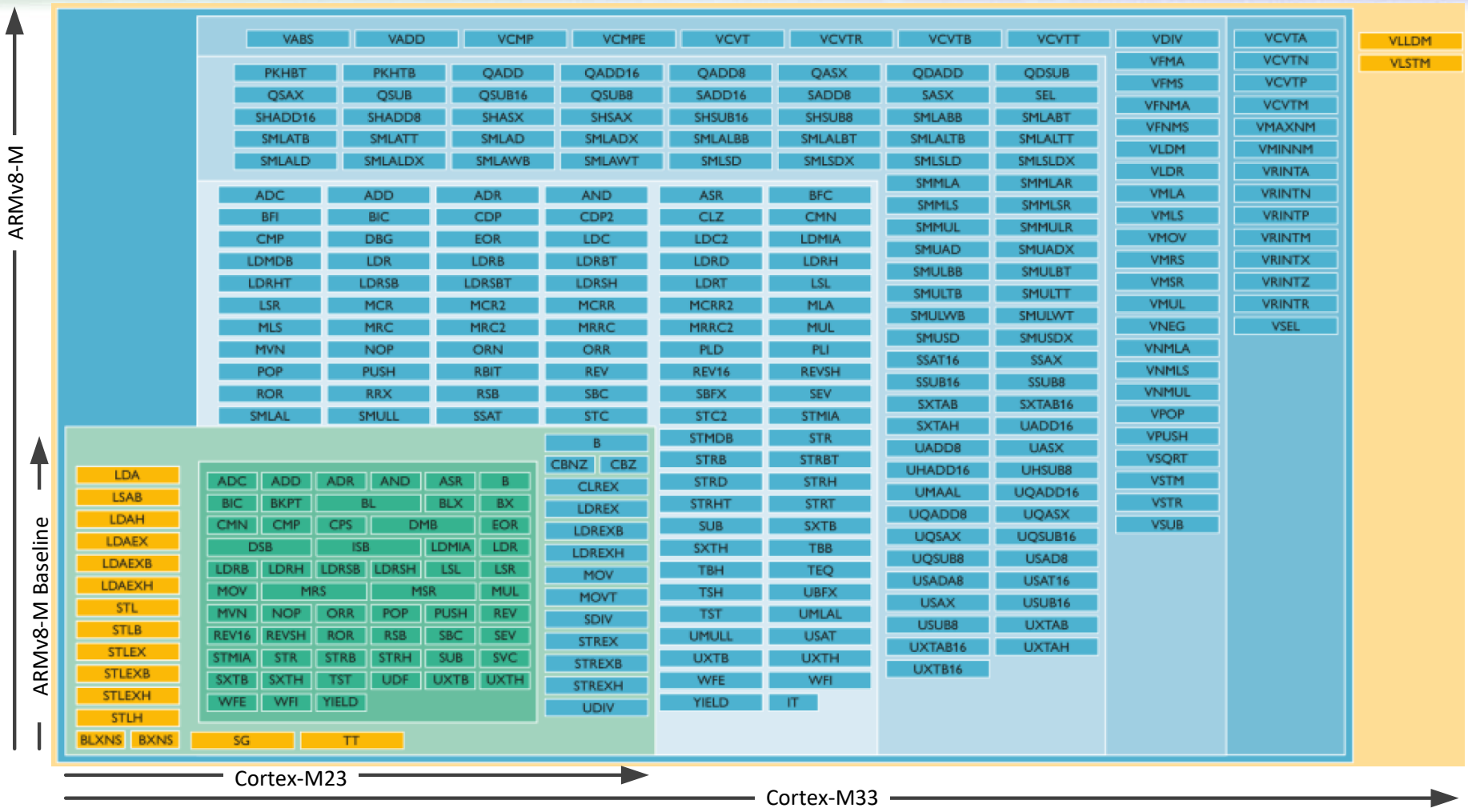
ARM Cortex M

Feature \ Core	M0 /M0+	M1	M3	M4	M7	M23	M33
ISA (ARMv...-M)	6	6	7	7E	7E	8, Baseline	8, Mainline
Thumb ISA (v4T, v5T, v6-M)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Thumb ISA (v7-M)	-	-	Yes	Yes	Yes	-	Yes
Low Power / Sleep mode (WFE, WFI, SEV)	Yes	NOP	Yes	Yes	Yes	Yes	Yes
Single cycle Multiply (32-bit result)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Bit field processing	-	-	Yes	Yes	Yes	-	Yes
Hardware divide (integer)	-	-	Yes	Yes	Yes	Yes	Yes
Unaligned data access	-	-	Yes	Yes	Yes	-	Yes
Table branch	-	-	Yes	Yes	Yes	-	Yes
Conditional execution (IT)	-	-	Yes	Yes	Yes	-	Yes
Compare & branch (CBZ, CBNZ)	-	-	Yes	Yes	Yes	Yes	Yes
Floating point	-	-	-	(Opt.) SP	SP, (Opt.) DP	-	(Opt.) SP
MAC	-	-	Multi-cycle, limited	Single-cycle	Single-cycle	-	Single-cycle
SIMD	-	-	-	Yes	Yes	-	Yes
Saturation	-	-	USAT, SSAT	Yes	Yes	-	Yes
Exclusive access	-	-	Yes	Yes	Yes	Yes	Yes
Load acquire, store release	-	-	-	-	-	Yes	Yes
Memory barrier	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SVC	Yes	Opt.	Yes	Yes	Yes	Yes	Yes
TrustZone support	-	-	-	-	-	Yes	Yes

Architecturi (ISA) – ARMv6-M, ARMv7-M



Architecturi (ISA) – ARMv8-M



ARM Cortex M – Referințe

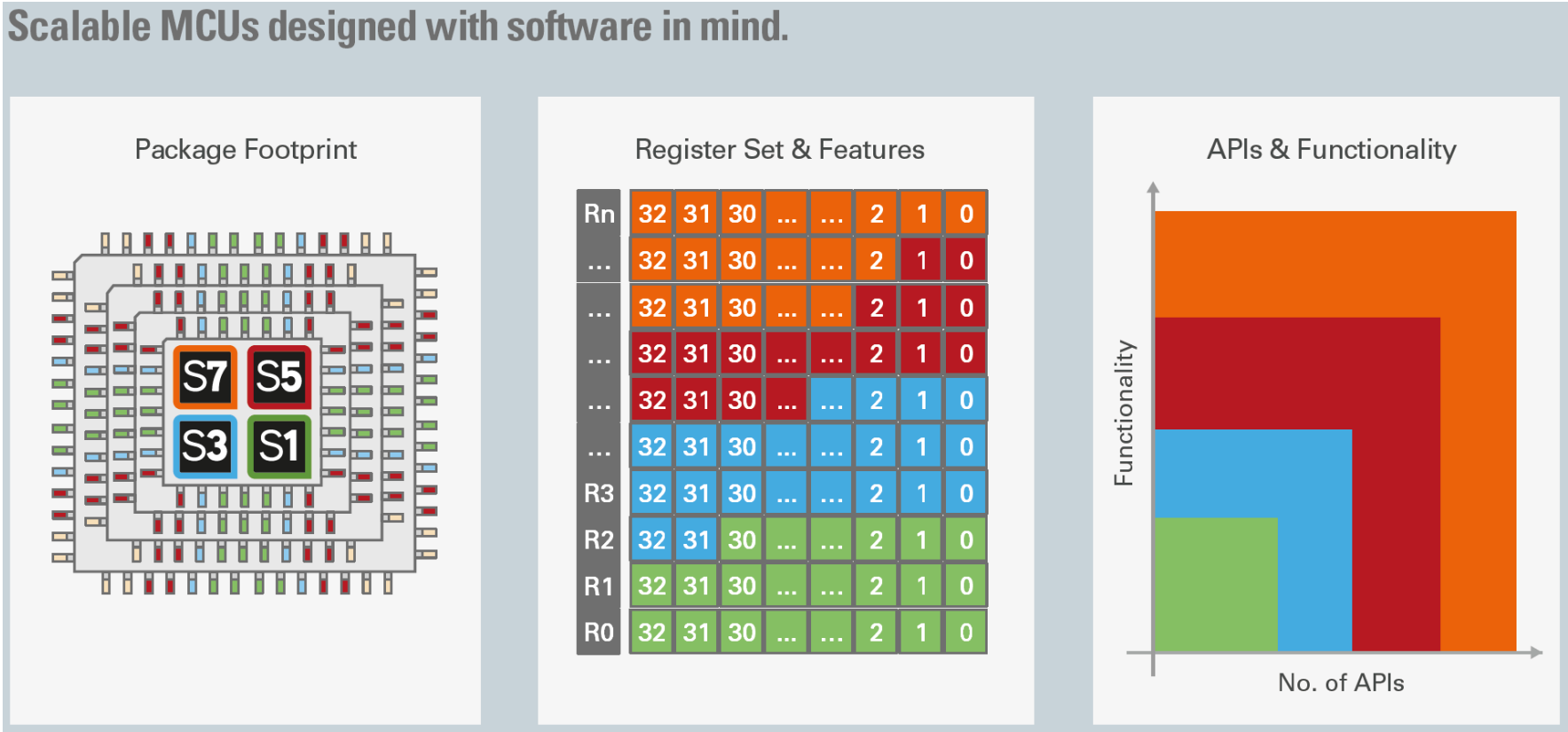
- https://community.arm.com/.../Cortex-M-resources
- https://community.arm.com/.../Cortex_M-for-Beginners-2017_v2.pdf
- https://developer.arm.com/

ARM Cortex-M

- exemplu de producător integrator -

ARMv8-M – Integrat în MCU: Renesas Synergy Platform MCUs

- **Scalabilitate (expandare), compatibilitate la nivel de capsulă (pini)**



ARM Cortex M in Renesas Synergy MCUs

- renesas.com/.../microcontrollers-microprocessors/renesas-synergy-platform-mcus

- Stiva -

- analiză comparativă -

Me: *Writing a recursive Fibonacci sequence*

My Stack Memory:

The magic of recursion

Stiva – Generalități

- Stiva = o zonă de memorie folosită convențional ca o structură LIFO
- Utilizată pentru:
  - **salvarea contextului**
  - **transmiterea parametrilor între funcții**
  - **stocarea adresei de întoarcere din funcție**
  - **stocarea temporară de date**
  - **stocarea variabilelor locale funcției**
- *Unelte* pentru lucrul cu stiva:
  - **elementare de interacțiune:**
    - PUSH: încărcare în stivă
    - POP: descărcare din stivă
  - **registre dedicate folosite de instrucțiunile de interacțiune**
- Baza(*bottom*) stivei = adresa de început a stivei

Intel 8086

CPU

8086 – Stiva – Caracteristici

- Stiva se populează (crește) **descrescător**
  - **PUSH realizează decrementarea SP**
  - **element introdus mai recent = adresă mai mică**

- SP – registrul Pointer de Stivă (*Stack Pointer*) (16 bit)
- SS – registrul Segment de Stivă (*Stack Segment*) (16 bit)

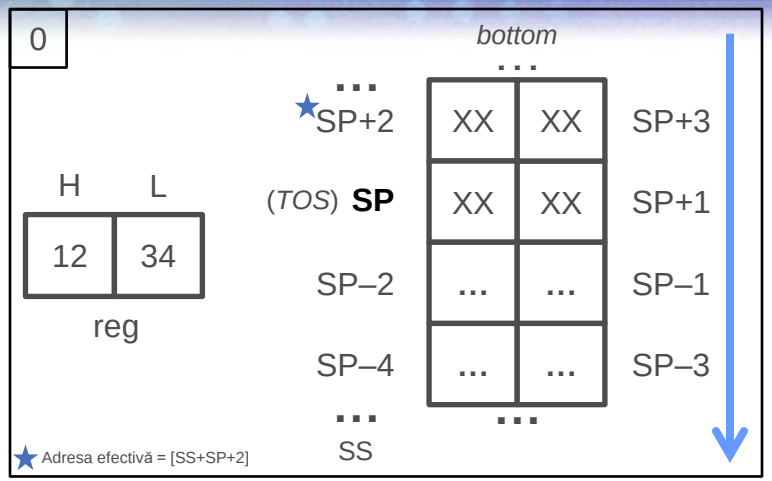
- SS<<4 + SP = adresa fizică a ultimului element introdus
  - **SS conține adresa segmentului de stivă, SP conține deplasament față de SS**

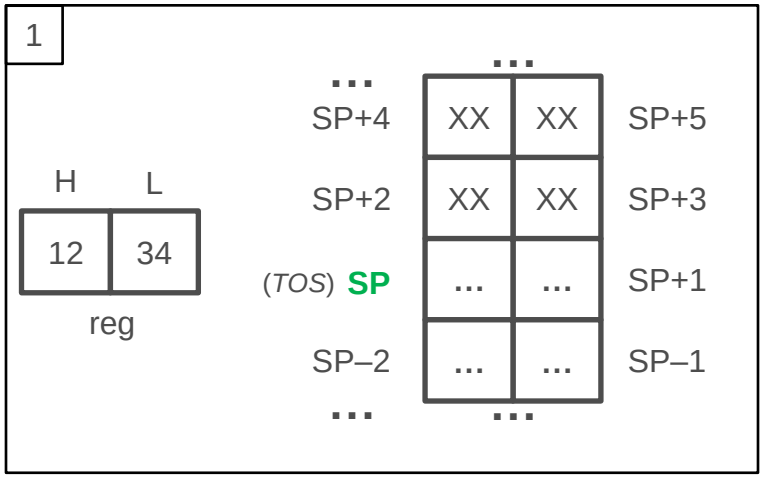
- POP, PUSH: doar cuvinte de 16 biți (2 octeți)

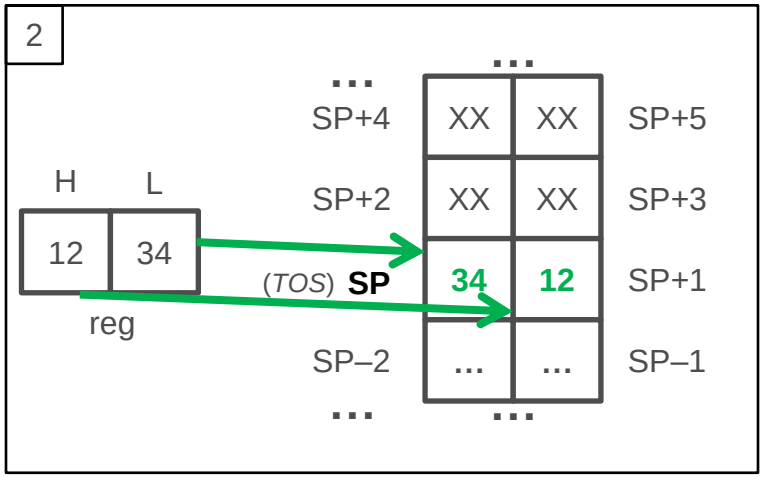
8086 – Încărcare în stivă – PUSH

- *TOS* = vârful stivei (*Top of Stack)*
- *bottom* = începutul stivei

- PUSH reg, echivalent cu:
  SP = SP – 2, SS:[SP] = reg



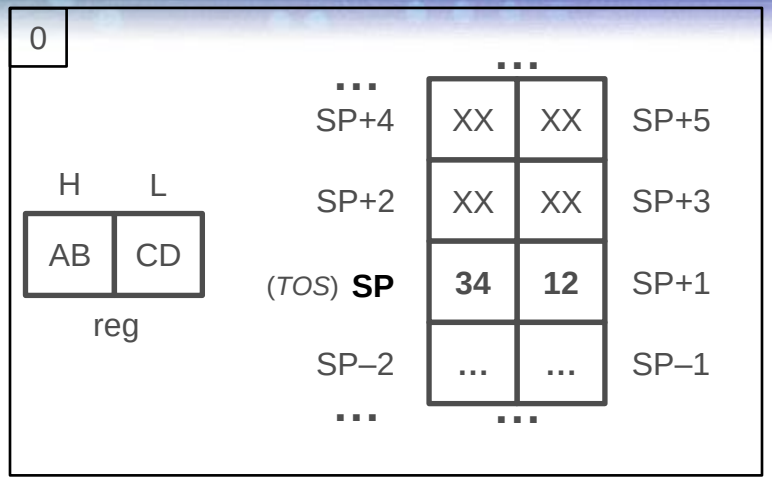


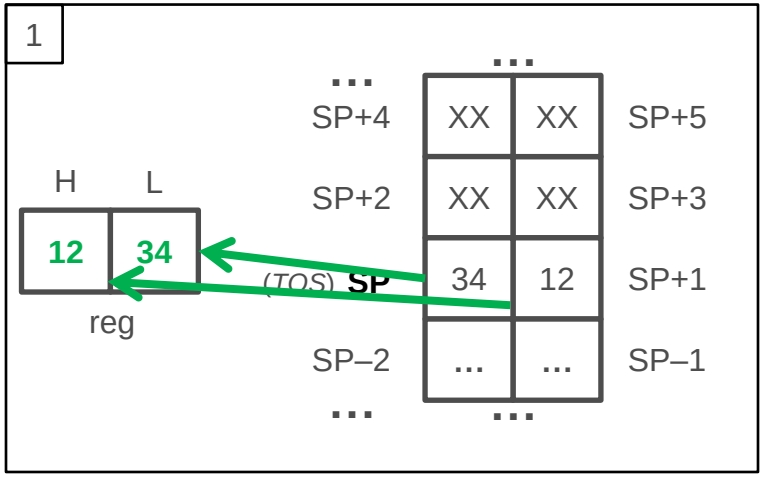


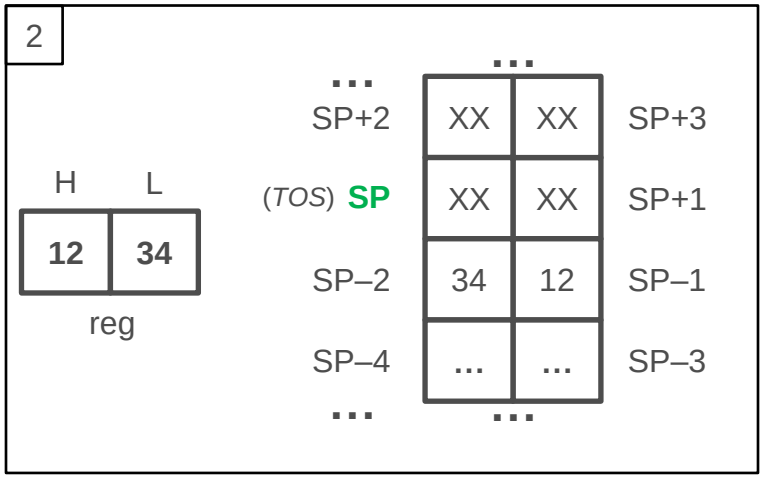
XX = date deja existente in stivă

8086 – Descărcare din stivă – POP

- **descărcare ≠ ștergere**
- POP reg, echivalent cu:
  reg = SS:[SP], SP = SP + 2;





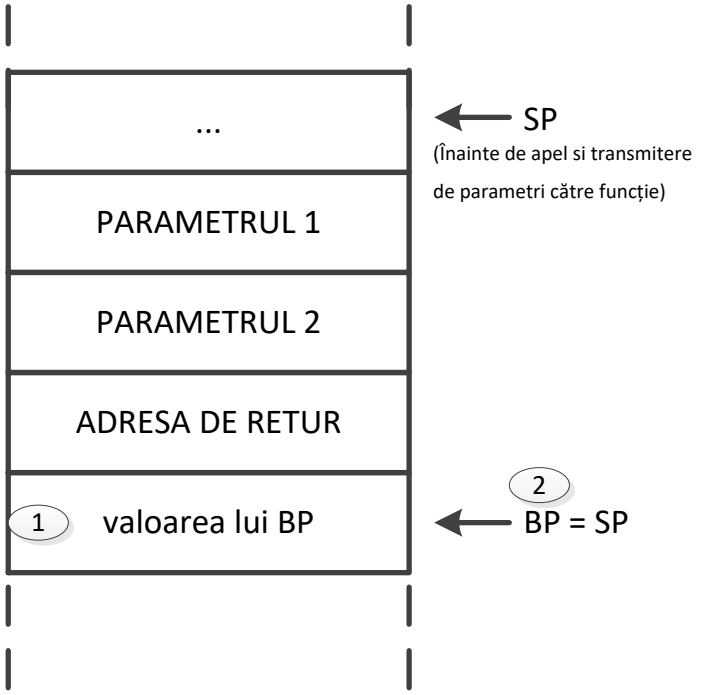


XX = date deja existente in stivă

8086 – Stiva – Caracteristici

- Stiva = o zonă din memoria RAM *(RAM organizată **Little-Endian** – în cazul 8086)*
  - **datele pot fi accesate și în alt mod decât prin POP**
    - adresare bazată sau adresare bazată indexată – BP ca registru bază

```
; suma a două numere
SUMA      PROC

PUSH      BP          (1)
MOV       BP, SP      (2)

; [BP+4] = PARAMETRUL 2
; [BP+6] = PARAMETRUL 1
MOV       AX, [BP+4]
ADD       AX, [BP+6]

POP       BP
RET       4

SUMA      ENDP
```



8086 – Stiva – Caracteristici

- Pot exista mai multe regiuni de memorie folosite pe post de stivă
  - **permite execuția "în paralel" a mai multor programe, fiecare cu segmentul său de stivă**
  - **selectarea stivei de folosit se face prin modificarea lui SS (stack segment)**
  - **o aplicație (un program) nu poate avea mai multe segmente de stivă**
- O stivă poate avea o dimensiune maximă de 64kB
  - **64kB = spațiul de memorie maxim adresabil pe 16 biți**

8086 – Stiva – Depășire

- PUSH când SP == 0 rezultă în depășirea capacității de stocare a stivei (stack overflow)
  - **efectul este suprascrierea datelor de la baza stivei (0-2 = FFFE)**

  (daca stiva este declarată de dimensiune maximă)

- 8086 nu are mecanism hardware de verificare a depășirii stivei
- Verificarea dacă stiva urmează a fi depășită trebuie făcută de către cel ce scrie programul, de exemplu:

```
        …
        CMP     SP, 0x0000
        JNE     eticheta_no_ovf     ; short-label jump (salt în același segment – intra-segment)
        JMP     eticheta_ovf        ; long-label jump (salt în alt segment – inter-segment)
eticheta_no_ovf:
        PUSH    …
        …
```

8086 – Stiva – Apel de funcție

- Instrucțiunea **CALL** – apel de funcție
  - **generează salvarea pe stivă a adresei de retur**
  - **parametrii funcției se recomandă a fi transmiși prin stivă**

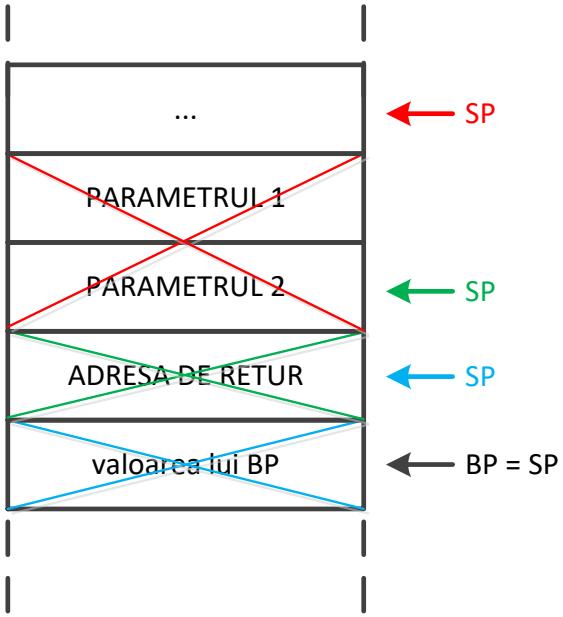
- Modul de compunere al adresei de retur diferă în funcție de adresa procedurii la care se va face apelul. Dacă adresa procedurii se află:
  - **în interiorul segmentului (*NEAR*) – se salvează doar IP (*Instruction Pointer*)**
  - **în exteriorul segmentului (*FAR*) – se salvează IP *și* CS (*Code Segment*)**
    - (segmentul de cod al programului apelant diferă de cel al funcției apelate)

- Saltul la vectorul de întrerupere se face prin instrucțiunea **CALL**
  - **fără transmitere de parametri**

8086 – Stiva – Retur din funcție

- Instrucțiunea **RET** (*return*)
  - **generează descărcarea de pe stivă și saltul la adresa de retur**
  - **utilizată împreună cu un operand influențează comportamentul lui SP**
    - operandul denotă numărul (par) de octeți din stivă peste care SP-ul "să sară"
  - **exemplu:**

```
; suma a două numere
SUMA    PROC

PUSH    BP
MOV     BP, SP

; [BP+4] = PARAMETRUL 2
; [BP+6] = PARAMETRUL 1
MOV     AX, [BP+4]
ADD     AX, [BP+6]

POP     BP
RET     4

SUMA    ENDP
```



8086 – Stiva – Retur din funcție

- Instrucțiunea **IRET** - revenire din rutina de tratare a întreruperii
  - ***interrupt return***
  - **descarcă din stivă registrul de stare (*FLAGS*)**
  - **descărcă din stivă registrele IP și CS**
  - **realizează saltul la CS:[IP], adică la adresa de revenire**

8086 – Stiva – Instrucțiuni de lucru **direct**

- Instrucțiunile **PUSHF, POPF** ***(F de la FLAGS)***
  - **încarcă / descarcă registrul de stare**

- Instrucțiunile **PUSHA, POPA** ***(A de la ALL)***
  - **încarcă / descarcă toate registrele procesorului: AX, CX, DX, BX, SP, BP, DI, SI**
  - **instrucțiune introdusă începând cu arhitectura succesoare (80186)**

8086 – Stiva – Instrucțiuni de lucru **direct**

MNEMONICĂ	NUMĂR DE CICLI PROCESOR *	Valabil la 8086	DESCRIERE
PUSH	8, 8, 17 + EA**	da	încărcare cuvânt (2 octeți) pe stivă
POP	11, 10, 16 + EA**	da	descărcare cuvânt de pe stivă
PUSHF	10	da	xxxx **O D I T S Z** x **A** x **P** x **C** ***
POPF	8	da	C, P, A, Z, S, T, I, D, O
PUSHA	17 (80286)	80x86	AX, CX, DX, BX, SP, BP, DI, SI
POPA	19 (80286)	80x86	SI, DI, BP, SP, BX, DX, CX, AX
…	…	80x86	…

*** pentru operand: registru, registru de segment, memorie**

**** număr de cicli în funcție de adresa efectivă**

***** x – valoare logică nedefinită**

8086 CPU – Referințe

- The 8086 Family User's Manual, Intel Corporation, October 1979
- The Intel Microprocessors 8th Edition, B.B. Brey, Pearson Education, 2009
- https://usermanual.wiki/Document/Intel8086FamilyUsersManual.2660397397/view
- https://userpages.umbc.edu/~squire/intel_book.pdf
- https://en.wikipedia.org/wiki/Intel_8086

Intel 8051 Core

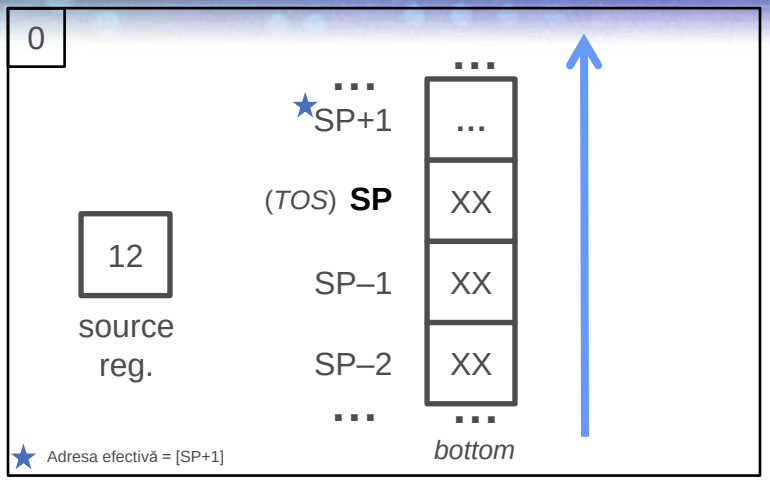
8051 – Stiva – Caracteristici

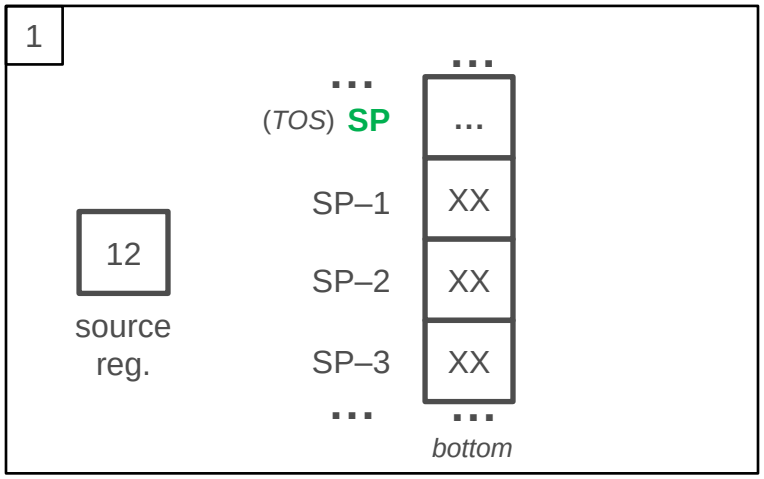
- Stiva se populează **crescător**
  - **PUSH realizează incrementarea SP**
  - **adresă mai mare = element introdus mai recent.**
- SP – registrul Pointer de Stivă (*Stack Pointer*) (8-bit)
  - **(inițializat după reset cu valoarea 0x08 – indică registrul R0 din grupul 2)**
  - **poate fi reinițializat să indice o altă adresă din RAM**
  - **își păstrează conținutul când procesorul intră în modul IDLE (consum redus)**
- POP, PUSH: doar câte 8 biți (1 octet)
- Datele pot fi accesate și prin adresare indirectă:
  - **scriere: MOV [SP+i], A ; i = deplasament (*offset*)**
  - **citire: MOV A, [SP–i] ; i = deplasament (*offset*)**

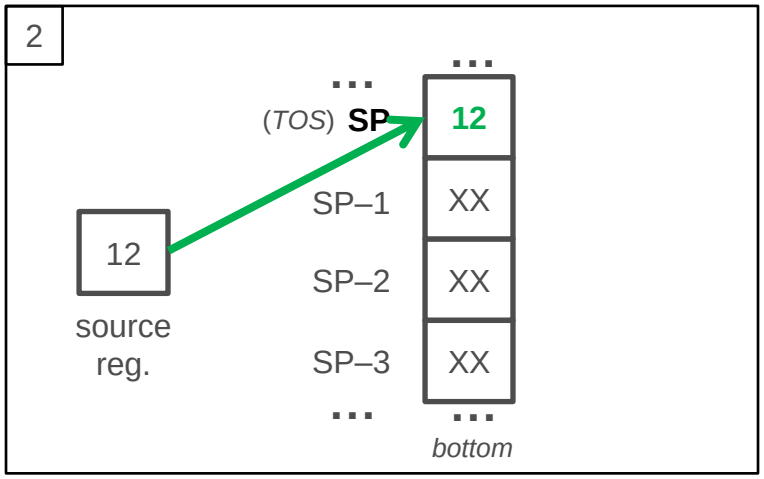
8051 – Încărcare în stivă – PUSH

- *TOS* = vârful stivei (*Top of Stack)*
- *bottom* = începutul stivei

- PUSH reg, echivalent cu:
  SP = SP + 1, [SP] = reg





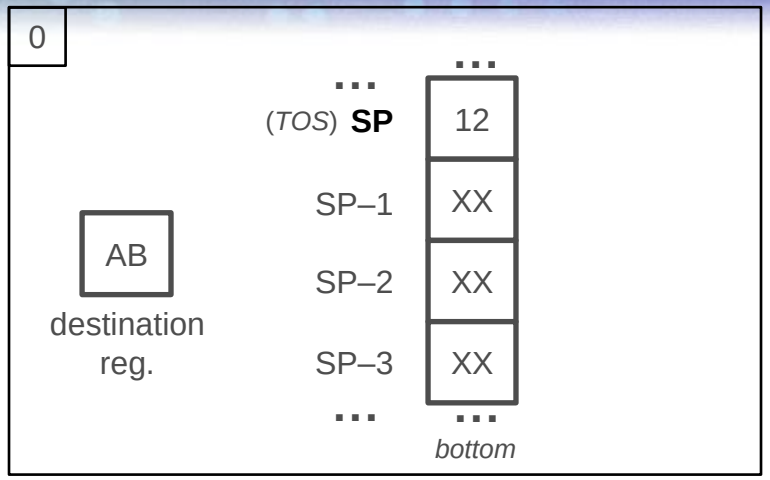


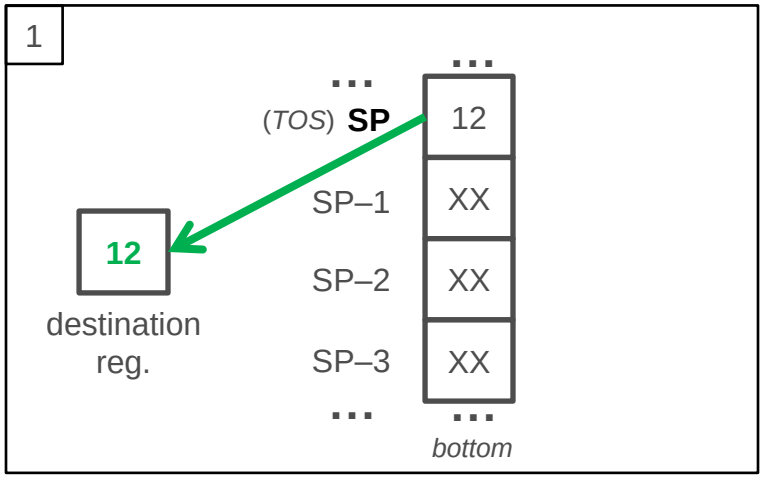
XX = date deja existente in stivă

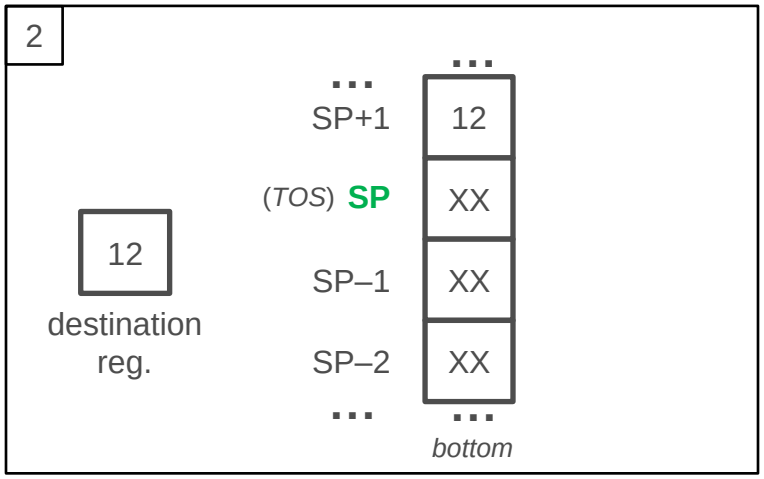
8051 – Descărcare din stivă – POP

- **descărcare ≠ ștergere**
- POP reg

  reg = [SP], SP = SP – 1;







XX = date deja existente in stivă

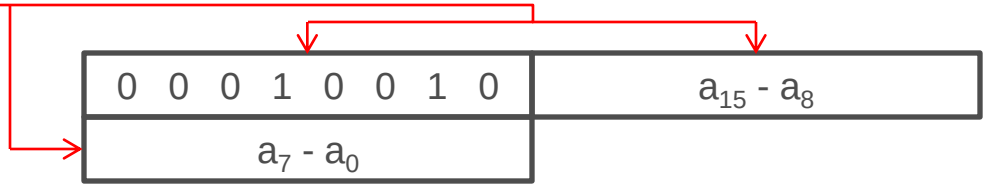
8051 – Stiva – Apel de funcție

- Instrucțiunea **ACALL** (*Absolute Call*) (codificare pe **2** octeți)
  - **apel de subrutină(funcție) într-un interval de maxim 2kB**
  - **operațiile efectuate microprogramat:**

PC = PC+2
SP = SP+1
[SP] = PC_{7-0}
SP = SP+1
[SP] = PC_{15-8}
PC_{10-0} = $addr_{10-0}$

- Instrucțiunea **LCALL** (*Long Call*) (codificare pe **3** octeți)
  - **apel de subrutină în întreg spațiul de 64kB**
  - **operațiile efectuate microprogramat:**

PC = PC+3
SP = SP+1
[SP] = PC_{7-0}
SP = SP+1
[SP] = PC_{15-8}
PC = $addr_{15-0}$



8051 – Stiva – Revenire din funcție

- Instrucțiunea **RET** (*Return from subroutine*)
  - **întoarcere din subrutină**
  - **operațiile efectuate microprogramat:**

PC_{15-8} = [SP]
SP = SP–1
PC_{7-0} = [SP]
SP = SP–1

0 0 1 0	0 0 1 0

- Instrucțiunea **RETI** (*Return from interrupt*)
  - **întoarcere din rutina de tratare a întreruperii**
  - **operațiile efectuate microprogramat:**

PC_{15-8} = [SP]
SP = SP–1
PC_{7-0} = [SP]
SP = SP–1

0 0 1 1	0 0 1 0

8051 – Stiva – Instrucțiuni de lucru

MNEMONICĂ	CICLI PROCESOR	Nume complet	DESCRIERE
PUSH	2	Push onto stack	încărcare cuvânt pe stivă
POP	2	Pop from stack	descărcare cuvânt de pe stivă
ACALL	2	Absolute call	apel de funcție / rutină
LCALL	2	Long call	apel de funcție / rutină
RET	2	Return	revenire din funcție
RETI	2	Return from interrupt	revenire din întrerupere

8051 Core – Referințe

- Intel MCS51 Microcontroller Family User's Manual, 1994
- http://web.mit.edu/6.115/www/document/8051.pdf
- https://en.wikipedia.org/wiki/Intel_MCS-51

Arhitecturile
ARMv4, ARMv5, ARMv6

ARMv5 – Stiva – Caracteristici

- R13 (SP) – registrul Pointer de Stivă (*Stack Pointer*) (32-bit)
  - **registru de uz general, cu scop dedicat de pointer de stivă**
- Stiva este aliniată la nivel de 4 octeți (32-bit)
- Fiecărui mod de execuție i se poate atribui o zonă de stivă

ARMv5 – Stiva – Caracteristici

- Stiva se populează *implicit* **descrescător "plin" (*full descending*)**
  - **implicit înseamnă utilizând instrucțiunile POP și PUSH**
  - **"plin" (full) înseamnă că SP indică la ultimul element introdus**
  - **"gol" (*empty*) înseamnă că SP indică următoarea locație liberă**
- *Explicit*, se poate alege modul de populare a stivei
  - **explicit înseamnă folosind instrucțiunile de tip LOAD–, STORE– Multiple**
  - **orice combinație de full / empty + ascending / descending este posibilă**

- **Modul de utilizare al stivei se precizează în setările compilatorului…**
  - **… pentru generarea instrucțiunilor corecte aferente modului de utilizare**

ARMv5 – Stiva – Instrucțiuni – Încărcare

Nume 1	Nume 2	Descriere Nume 1	Descriere Nume 2	Tip stivă
STMIB	STMFA	Inc. Before	Full Ascen.	Crescătoare, SP indică ultimul element introdus
STMIA	STMEA	Inc. After	Empty Ascen.	Crescătoare, SP indică primul loc liber
STMDB	STMFD	Dec. Before	Full Descen.	Descrescătoare, SP indică ultimul element introdus
STMDA	STMED	Dec. After	Empty Descen.	Descrescătoare, SP indică primul loc liber

Precizări:

- STM = Store Multiple
- Nume 2 = nume echivalent
- Inc. = Increment
- Dec. = Decrement
- Ascen. = Ascending
- Descen. = Descending

Pentru evitarea posibilelor erori umane, au fost create aceste “alias-uri” pentru instrucțiunile de lucru cu stiva. (Nume 2)

ARMv5 – Stiva – Instrucțiuni – Descărcare

Nume 1	Nume 2	Descriere Nume 1	Descriere Nume 2	Tip stivă
LDMIB	LDMFA	Inc. Before	Full Ascen.	Crescătoare, SP indică ultimul element introdus
LDMIA	LDMEA	Inc. After	Empty Ascen.	Crescătoare, SP indică primul loc liber
LDMDB	LDMFD	Dec. Before	Full Descen.	Descrescătoare, SP indică ultimul element introdus
LDMDA	LDMED	Dec. After	Empty Descen.	Descrescătoare, SP indică primul loc liber

Descărcare:
Identic ca la încărcare…?

ARMv5 – Stiva – Instrucțiuni – Descărcare

Nume 1	Nume 2	Descriere Nume 1	Descriere Nume 2	Tip stivă
LDMIB	LDMFA	Inc. Before	Full Ascen.	Crescătoare, SP indică ultimul element introdus
LDMIA	LDMEA	Inc. After	Empty Ascen.	Crescătoare, SP indică primul loc liber
LDMDB	LDMFD	Dec. Before	Full Descen.	Descrescătoare, SP indică ultimul element introdus
LDMDA	LDMED	Dec. After	Empty Descen.	Descrescătoare, SP indică primul loc liber

Descărcare:
Identic ca la încărcare…?

NU.

ARMv5 – Stiva – Instrucțiuni – Descărcare

Nume 1	Nume 2	Descriere Nume 1	Descriere Nume 2	Tip stivă
LDMIB	LDMFA	Inc. Before	Full Ascen.	Crescătoare, SP indică ultimul element introdus
LDMIA	LDMEA	Inc. After	Empty Ascen.	Crescătoare, SP indică primul loc liber
LDMDB	LDMFD	Dec. Before	Full Descen.	Descrescătoare, SP indică ultimul element introdus
LDMDA	LDMED	Dec. After	Empty Descen.	Descrescătoare, SP indică primul loc liber

Operațiile de încarcare și descărcare sunt complementare.
Deci și raționamentul este complementar.

ARMv5 – Stiva – Instrucțiuni – Descărcare

Nume 1	Nume 2	Descriere Nume 1	Descriere Nume 2	Tip stivă
LDMDA	LDMFA	Dec. After	Full Ascen.	Crescătoare, SP indică ultimul element introdus
LDMDB	LDMEA	Dec. Before	Empty Ascen.	Crescătoare, SP indică primul loc liber
LDMIA	LDMFD	Inc. After	Full Descen.	Descrescătoare, SP indică ultimul element introdus
LDMIB	LDMED	Inc. Before	Empty Descen.	Descrescătoare, SP indică primul loc liber

Precizări:

- LDM = Load Multiple
- Nume 2 = nume echivalent
- Inc. = Increment
- Dec. = Decrement
- Ascen. = Ascending
- Descen. = Descending

ARMv5 – Stiva – Instrucțiuni pereche

Nume 1 Încărcare	Nume 1 Descărcare	Nume 2 Încărcare	Nume 2 Descărcare	Tip stivă
STMIB	LDMDA	STMFA	LDMFA	Crescătoare, SP indică ultimul element introdus
STMIA	LDMDB	STMEA	LDMEA	Crescătoare, SP indică primul loc liber
STMDB	LDMIA	STMFD	LDMFD	Descrescătoare, SP indică ultimul element introdus
STMDA	LDMIB	STMED	LDMED	Descrescătoare, SP indică primul loc liber

Se observă complementaritatea operațiilor de: incrementare, respectiv decrementare și a momentului operației: înainte, respectiv după.

ARMv5 – Stiva – Load- / Store- Multiple

- Instrucțiunile Store / Load Multiple, indiferent de tipul de stivă:
  - **se pot utiliza și cu un singur registru**
    - exemplu:

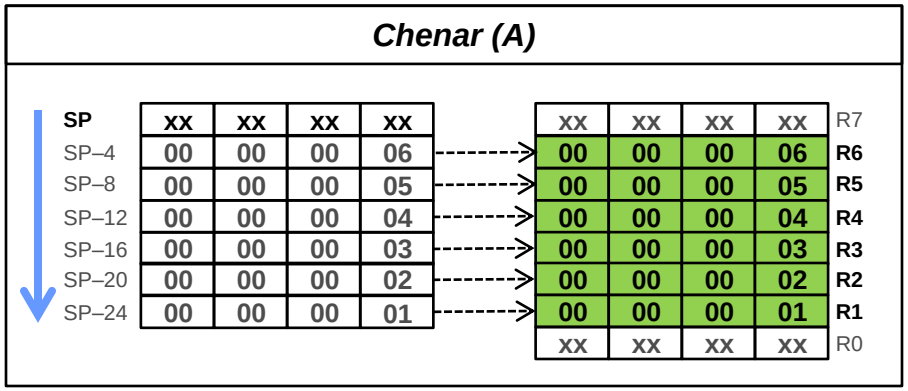
```
STMFD     SP!, {R1}          ; încarcă valoarea registrului R1 pe stivă
LDMFD     SP!, {R2}          ; descarcă valoarea din vârful stivei în R2
; rezultat: R2 = R1
```

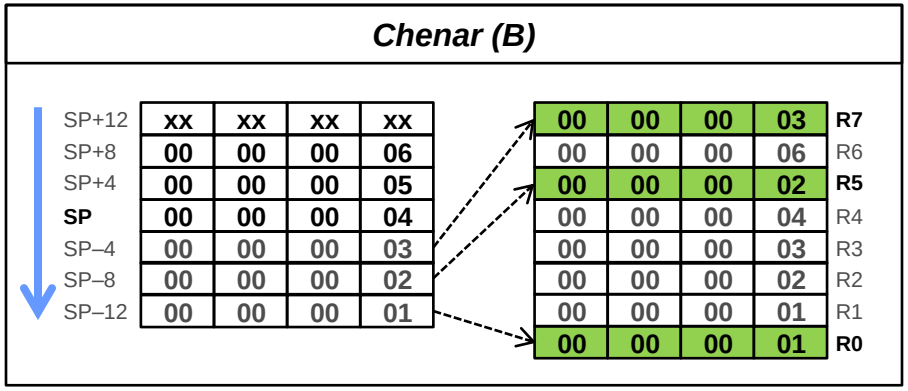
  - **ordinea încărcării / descărcării în / din stivă a registrelor se face în ordinea indexului fiecărui registru.**
    - exemplu:

```
STMFD     SP!, {R5, R1, R6, R2-R4}
; stiva ar fi arătat la fel daca s-ar fi folosit în loc instrucțiunea:
STMFD     SP!, {R2-R4, R5, R6, R1} (sau orice altă ordine a acestor registre)
```

ARMv5 – Stiva – Load- / Store- Multiple - Exemplu

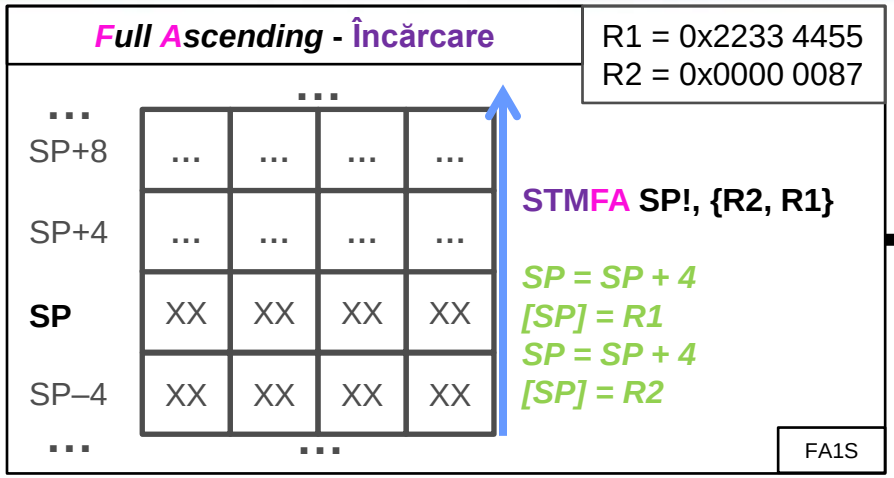
```
; inițializare registre
LDR         R1, =1
LDR         R2, =2
LDR         R3, =3
LDR         R4, =4
LDR         R5, =5
LDR         R6, =6
; încărcare în stivă
STMFD       SP!, {R5, R1, R6, R2-R4}
; descărcare din stivă (1)
LDMFD       SP!, {R6, R1-R5}
; …vezi rezultatele în Chenar (A)…

; reîncărcare registre în stivă
STMFD       SP!, {R1-R6}
; descărcare din stivă (2)
LDMFD       SP!, {R5, R0, R7}
; …vezi rezultatele în Chenar (B)…
```

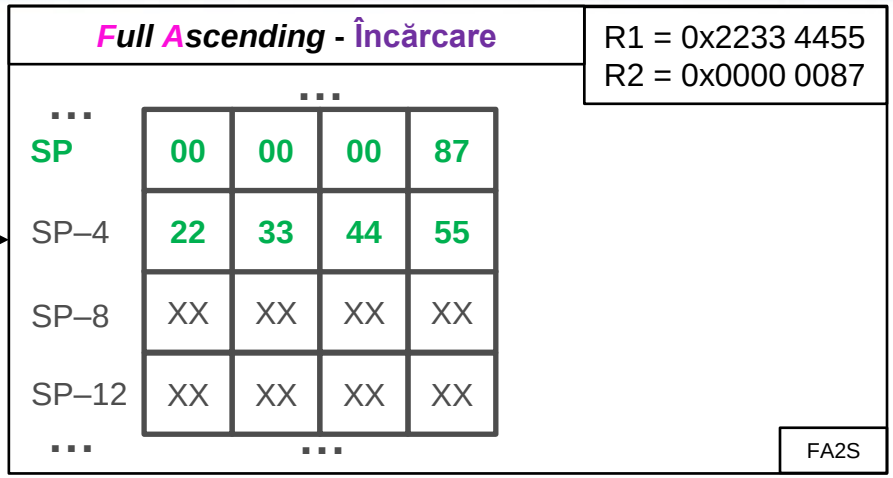


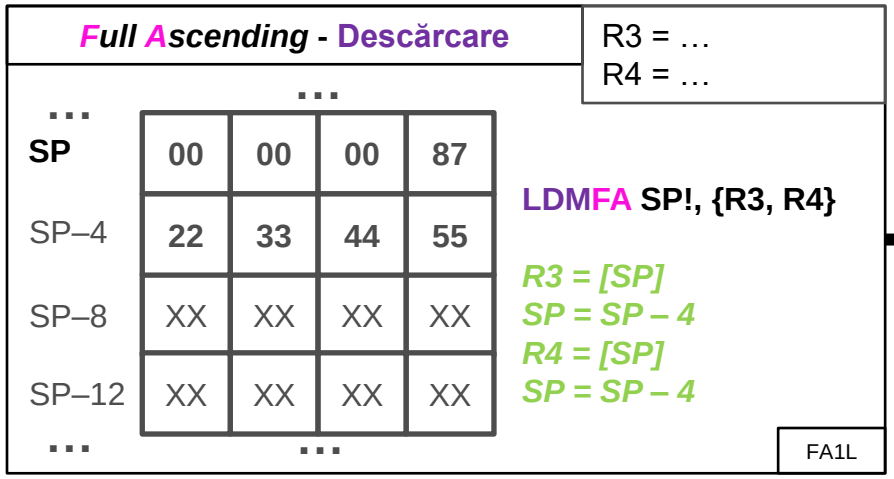


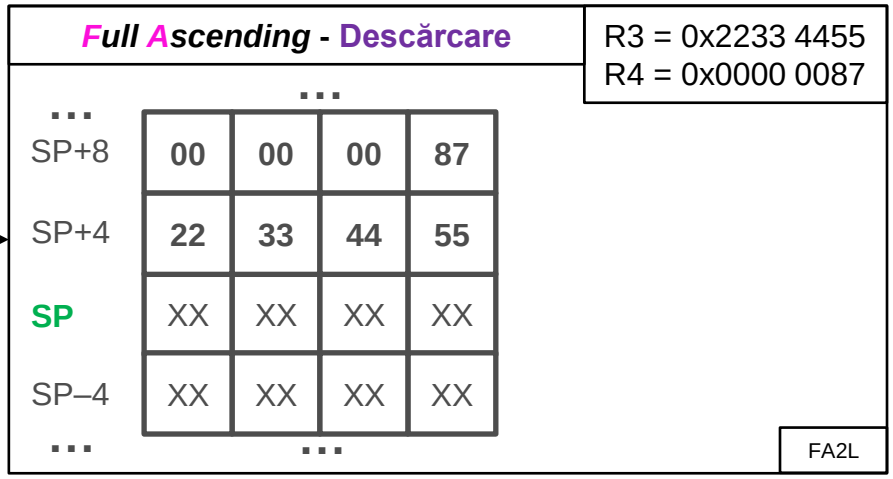
xx = date existente anterior, valoarea nu prezintă interes

ARMv5 – Stiva – Funcționare – *Full Ascending*



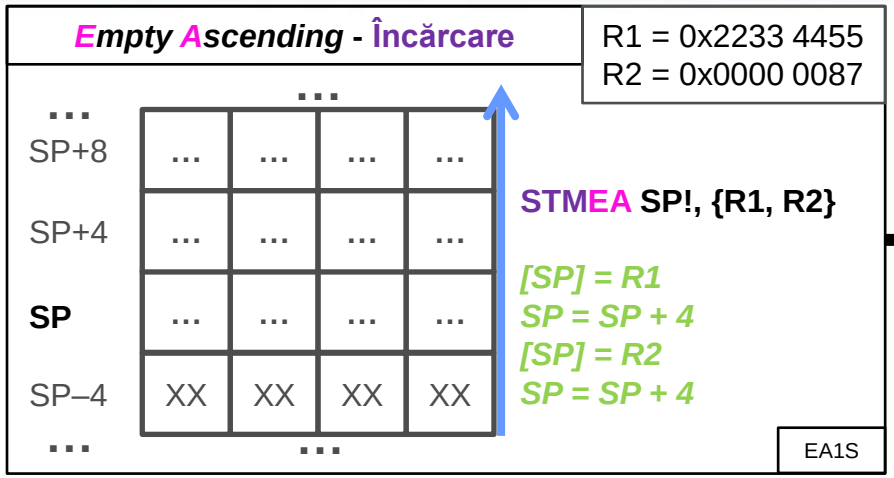


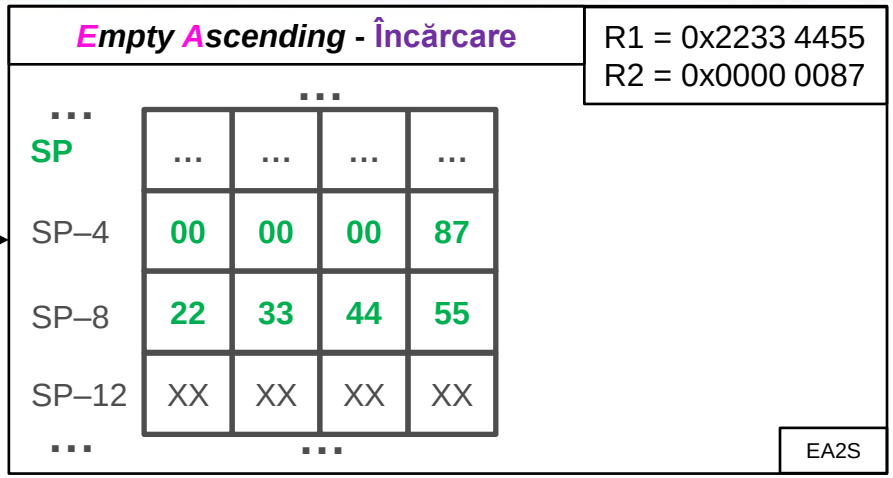


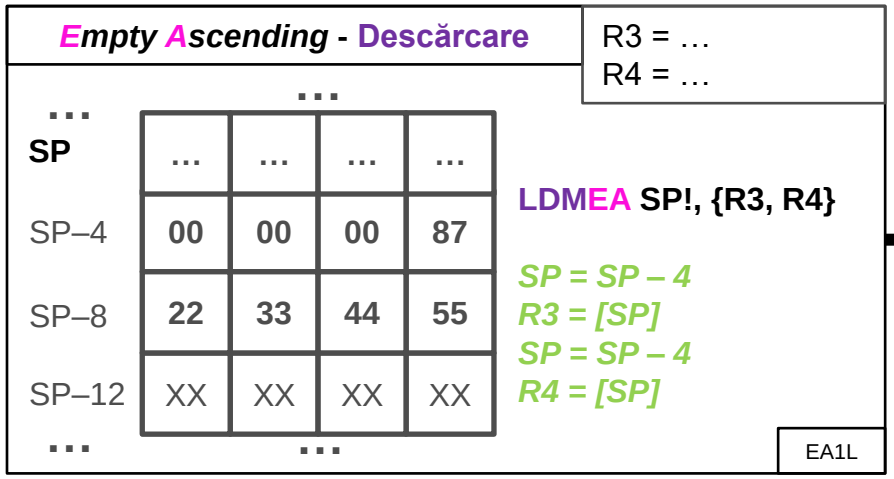


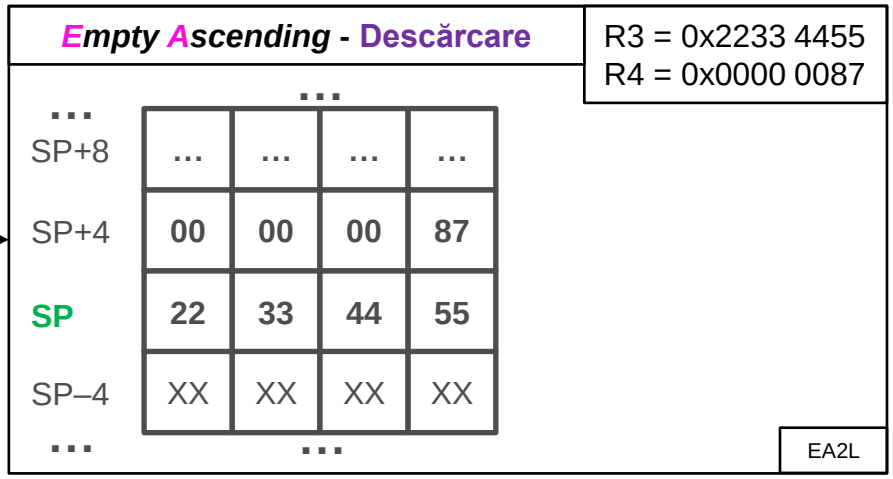
XX = date existente anterior in stivă

ARMv5 – Stiva – Funcționare – *Empty Ascending*



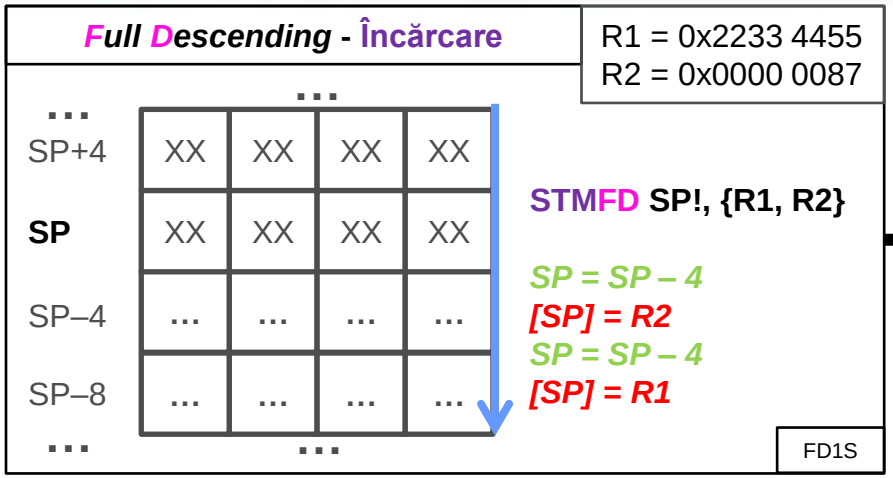


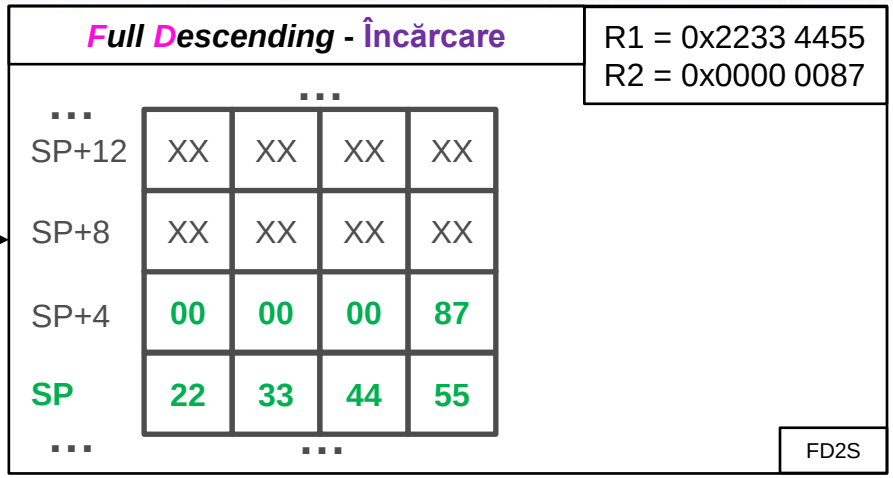


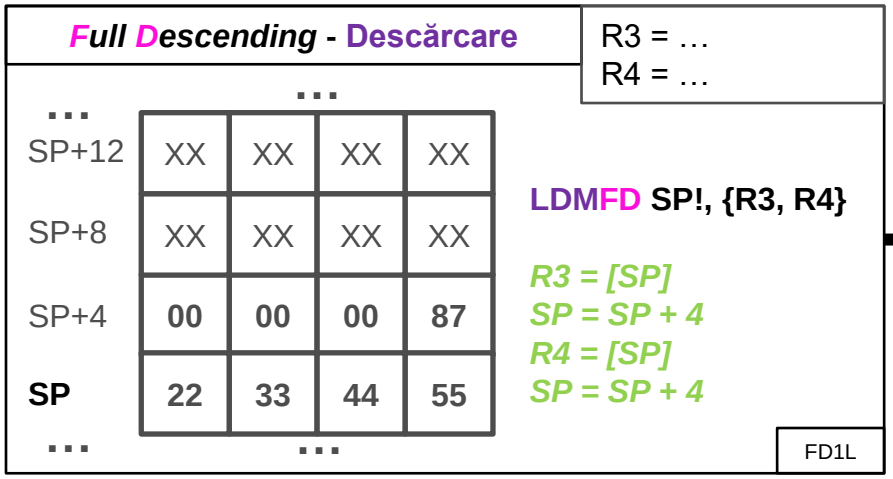


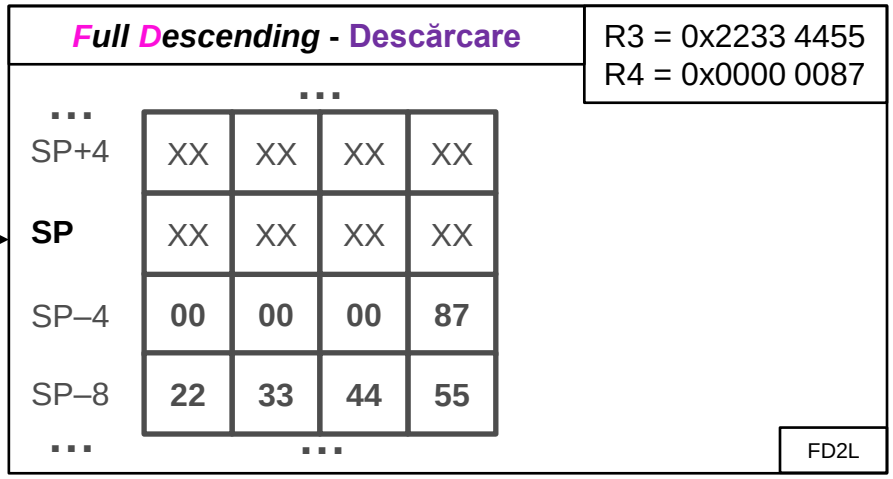
XX = date existente anterior in stivă

ARMv5 – Stiva – Funcționare – *Full Descending*



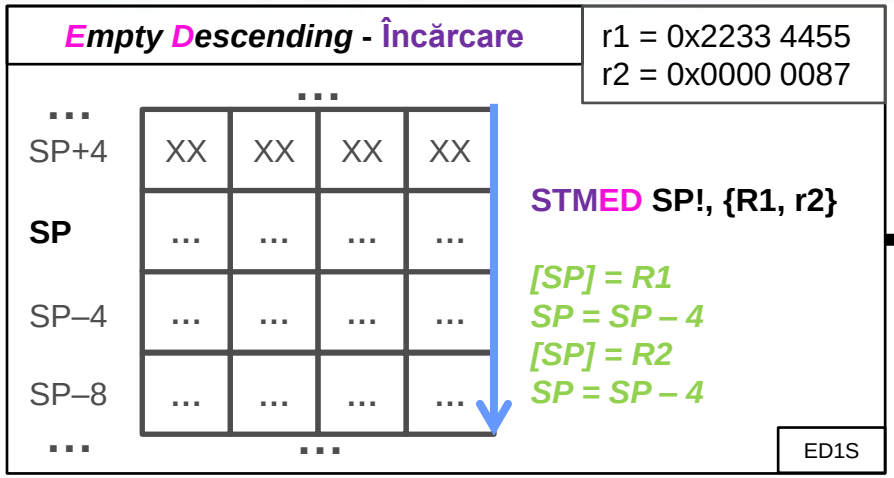


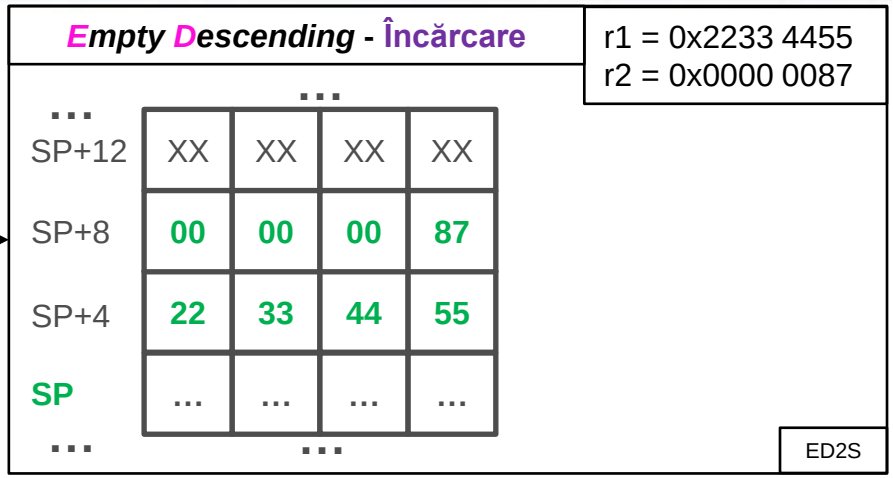


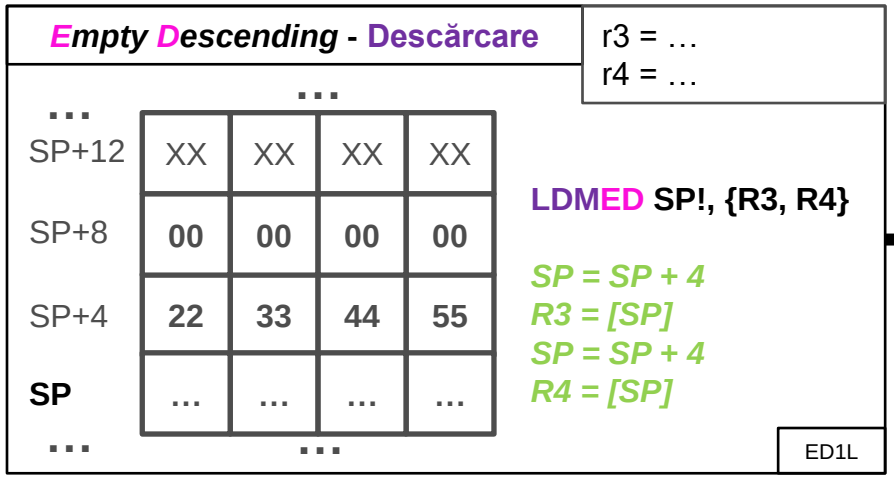


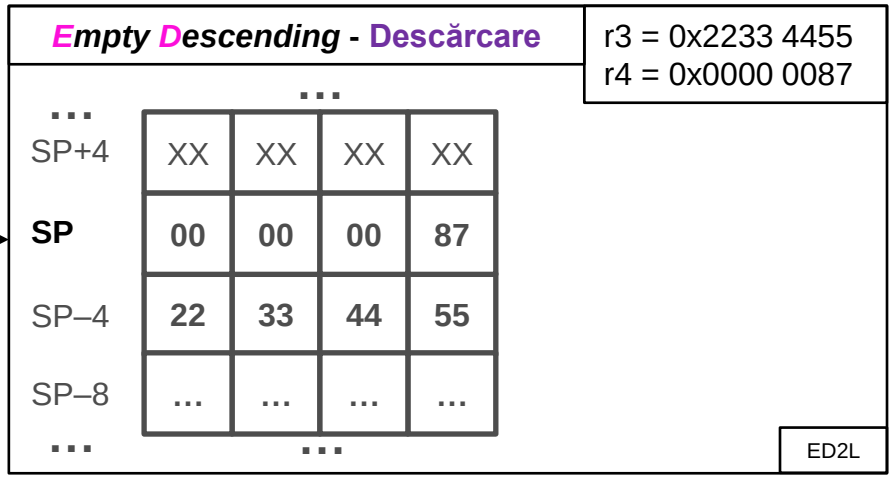
XX = date existente anterior in stivă

ARMv5 – Stiva – Funcționare – *Empty Descending*









XX = date existente anterior in stivă

ARMv5 – Stiva – Apel de funcție

- **BL** *etichetă*
  - creare legătură de retur și salt (***Branch*** *&* ***Link***)
  - ***etichetă*** este un deplasament (***offset***) față de PC
- **BL** *etichetă*
  - LR = PC + (T == 1? 2, 4) , PC = ***etichetă***

- **BLX** *etichetă* sau BLX *reg* (***Branch*** *&* ***Link*** *&* ***eXchange***)
  - creare legătură de retur și salt și specificare tip instrucțiuni: Thumb / ARM
  - ***etichetă*** este un deplasament (***offset***) față de PC
  - ***reg*** conține o adresă absolută
- **BLX** *etichetă*
  - LR = PC + (T == 1? 2, 4) , PC = ***etichetă***, T = 1
- **BLX** *reg*
  - LR = PC + (T == 1? 2, 4) , PC = ***reg*** & 0xFFFF FFFE, T = ***reg*** & 0x1

ARMv5 – Stiva – Apel de funcție

- **Utilizarea stivei în apelul funcției**
  - se vor salva pe stivă registrele care:
    - în funcția **apelantă** sunt utilizate la calcule, dar care se vor folosi pentru transmiterea de parametri către funcția **apelată**
    - sunt utilizate și in funcția **apelantă**, dar și in cea **apelată**

```
STMFD     SP!, {R0}    ; R0 va fi folosit pentru transmiterea unui parametru
LDR       R0, param1             ; se încarcă valoarea parametrului în R0
BL        nume_funcție
LDMFD     SP!, {R0}    ; refacerea lui R0 ; aici se revine din funcție
…

…
nume_funcție:
STMFD     SP!, {LR, R1}
… ; operații ce implică și utilizarea lui R1
LDMFD SP!, {PC, R1}  ; refacerea lui R1 și revenirea din funcție
```

ARMv5 – Stiva – Instrucțiuni Thumb

- Thumb – setul redus de instrucțiuni codificate 16-bit
  - **doar PUSH, POP (similare instrucțiunilor STMFD și LDMFD)**
    - doar R0-R7 și LR pot încărcate / descărcate (din cauza codificării 16-bit)
  - **instrucțiunile PUSH, POP tratează stiva ca fiind una *Full Descending***

- ARM-Thumb Procedure Call Standard (ATPCS)
  - **stabilește cum se desfășoară un apel de funcție:**
    - modul de utilizare al stivei (= full descending)
    - modul de utilizare al registrelor

- https://developer.arm.com/documentation/dui0056/d/using-the-procedure-call-standard/about-the-arm-thumb-procedure-call-standard

ARM7TDMI – Stiva

MNEMONICĂ	CICLI PROCESOR	Nume complet	DESCRIERE
PUSH	(n-1)S + 2N	Push multiple registers	Încarcă unul sau mai multe registre în stivă
STMxy	(n-1)S + 2N	Store multiple registers	Stochează unul sau mai multe registre începând de la o adresă
POP	nS + N + 1	Pop multiple registers	Descarcă unul sau mai multe registre din stivă
LDMxy	nS + N + 1	Load multiple registers	Citește una sau mai multe valori începând de la o adresă și stochează în registre

xy = orice combinație de F sau E cu A sau D
n = numărul de registre
N = ciclu de magistrală – acces non-secvențial
S = ciclu de magistrală – acces secvențial

ARMv5 – Stiva – Referințe

- ARM DDI 0100I - ARM Architecture Reference Manual, 2005
- ARM DDI 0210C - ARM7 TDMI Technical Reference Manual, R4p1, 2004
- ARM System Developer's Guide – Designing and Optimizing System Software, A. N. Sloss, D. Symes, C. Wright, Elsevier MK, 2004
- UM10237 – NXP LPC24XX User manual, Rev. 04 – 2009
- https://developer.arm.com/ip-products/processors/classic-processors
- **VisUAL - ARM Emulator - Editor & Assembler:**

  https://salmanarif.bitbucket.io/visual/downloads.html

ARMv8-M Architecture

ARMv8-M – Stiva – Caracteristici

- R13 (SP) – registrul Pointer de Stivă (*Stack Pointer*) (32-bit)
  - **registru de uz general, cu scop dedicat de pointer de stivă**
- Stiva este aliniată la nivel de 4 octeți (32-bit)
- Fiecărui mod de execuție i se poate atribui o zonă de stivă
- Există registru dedicat pentru detecția depășirii stivei - *stack limit register*
  - **declanșează o excepție (UsageFault, stack overflow)**
  - **valabil doar la versiunea Mainline**

ARMv8-M – Stiva – Caracteristici

- Dacă extensia de securitate (*Security Extension*) este implementată:
  - **există câte două registre SP pentru fiecare dintre cele doua stări de securitate**
  - **există câte două *stack limit register* pentru fiecare dintre cele doua stări de securitate**
  - **după reset, MSP_S este cel inițializat și folosit**

Tip stivă	Stiva	Pointer de stivă	*Stack limit register*
Securizată (*Secure*)	**Main**	**MSP_S**	**MSPLIM_S**
	Procesului	**PSP_S**	**PSPLIM_S**
Nesecurizată (*Non-Secure*)	**Main**	**MSP_NS**	**MSPLIM_NS**
	Procesului	**PSP_NS**	**PSPLIM_NS**

ARMv8-M – Stiva – Caracteristici

- Daca extensia de securitate nu este implementată:
  - **există doua registre SP**
  - **există două *stack limit register***
  - **după reset, MSP este cel inițializat și folosit**

Tip stivă	Stiva	Pointer de stivă	*Stack limit register*
---	Main	MSP	MSPLIM
	Procesului	PSP	PSPLIM

ARMv8-M – Stiva – Integritate

- Stack overflow – verificare hardware
  - **generează excepția UsageFault**
- Semnătura de integritate
  - **mecanism disponibil doar în implementarea cu *Security Extension***
  - **adițional datelor încărcate, se salvează pe stivă și o valoare CRC-32 calculată pe baza datelor încărcate**
  - **la refacerea contextului, se recalculează valoarea CRC-32 pe baza datelor descărcate și se verifică egalitatea cu semnătura descărcată**
  - **generează excepția *SecureFault* dacă semnăturile nu sunt egale**
    - (adică datele descărcate nu corespund cu datele anterior încărcate, sau semnătura încărcată nu corespunde cu semnătura calculată după descărcarea datelor)

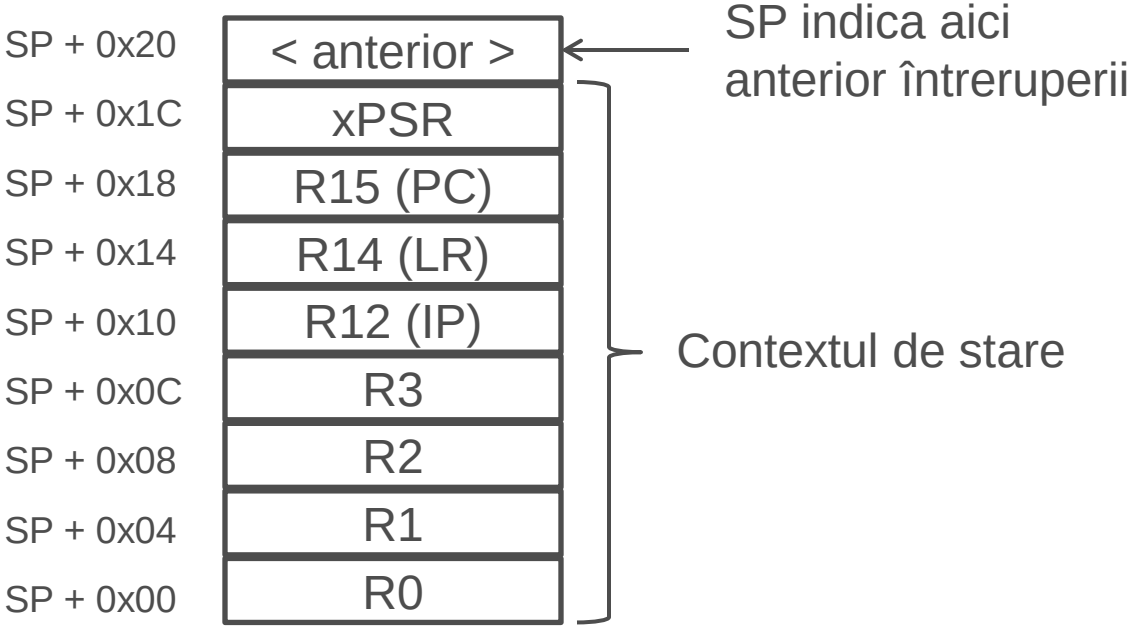
ARMv8-M – Cortex M23 – Stivă – cadrul scurt

R0 – R3: registre de lucru (folosite în instrucțiuni)

IP – intra-procedure-call register: registru pentru transmisia de date între rutină și o subrutină de-a sa.

LR – Link Register – conține o valoare de tip **EXC_RETURN**

PC – Program Counter

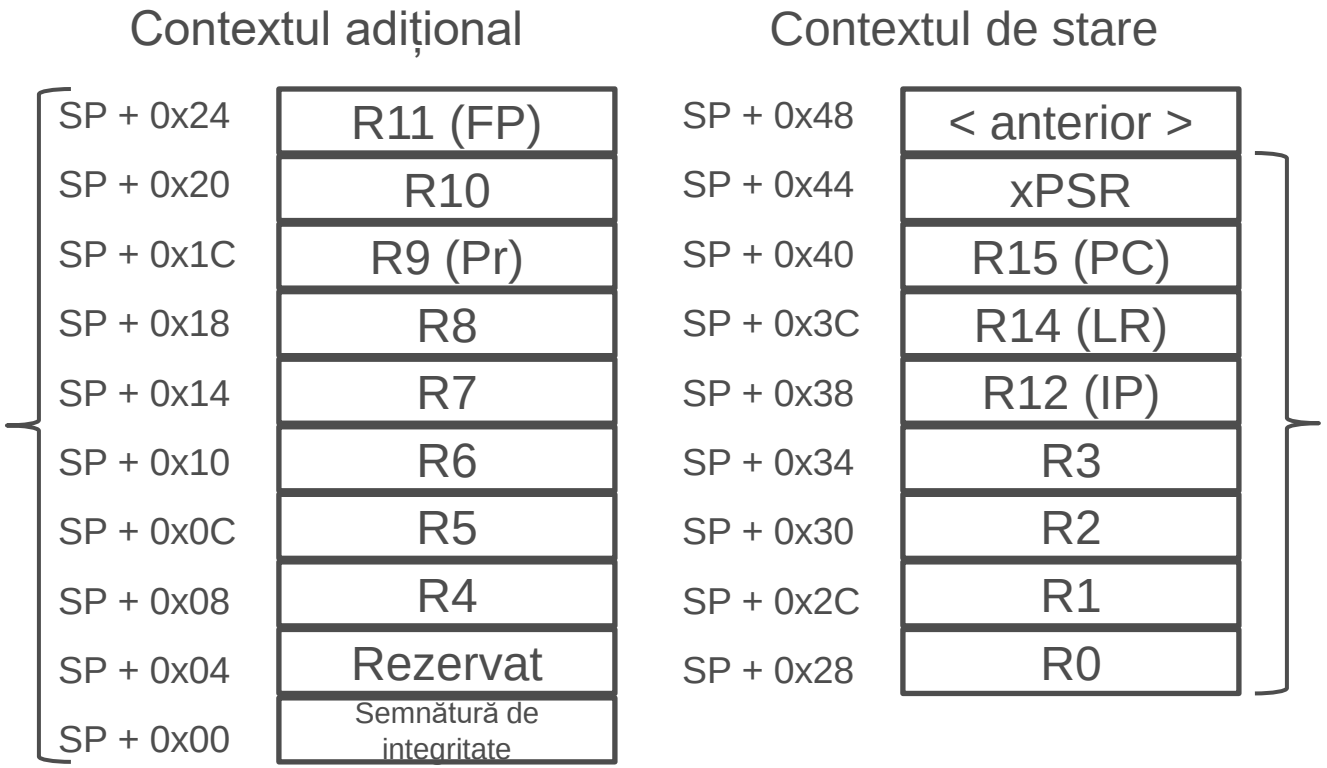


ARMv8-M – Cortex M23 – Stivă – cadrul extins

R4 – R8, R10, R9: folosite la stocarea variabilelor locale

Pr – Platform register – funcționalitate dependentă de platformă

FP – Frame Pointer – indică stiva folosită la intoarcere



Contextul aditional:
salvat atunci când se întrerupe execuția securizată, pentru a trata o excepție configurată ca fiind nesecurizată

ARMv8-M – Cortex M23

- Cadrul scurt este utilizat atunci când:
  - **intrarea se face din starea *Non-secure***
  - **sau opțiunea *Security extension* nu este implementată**
  - **sau utilizarea *extended stack frame* nu este necesară**

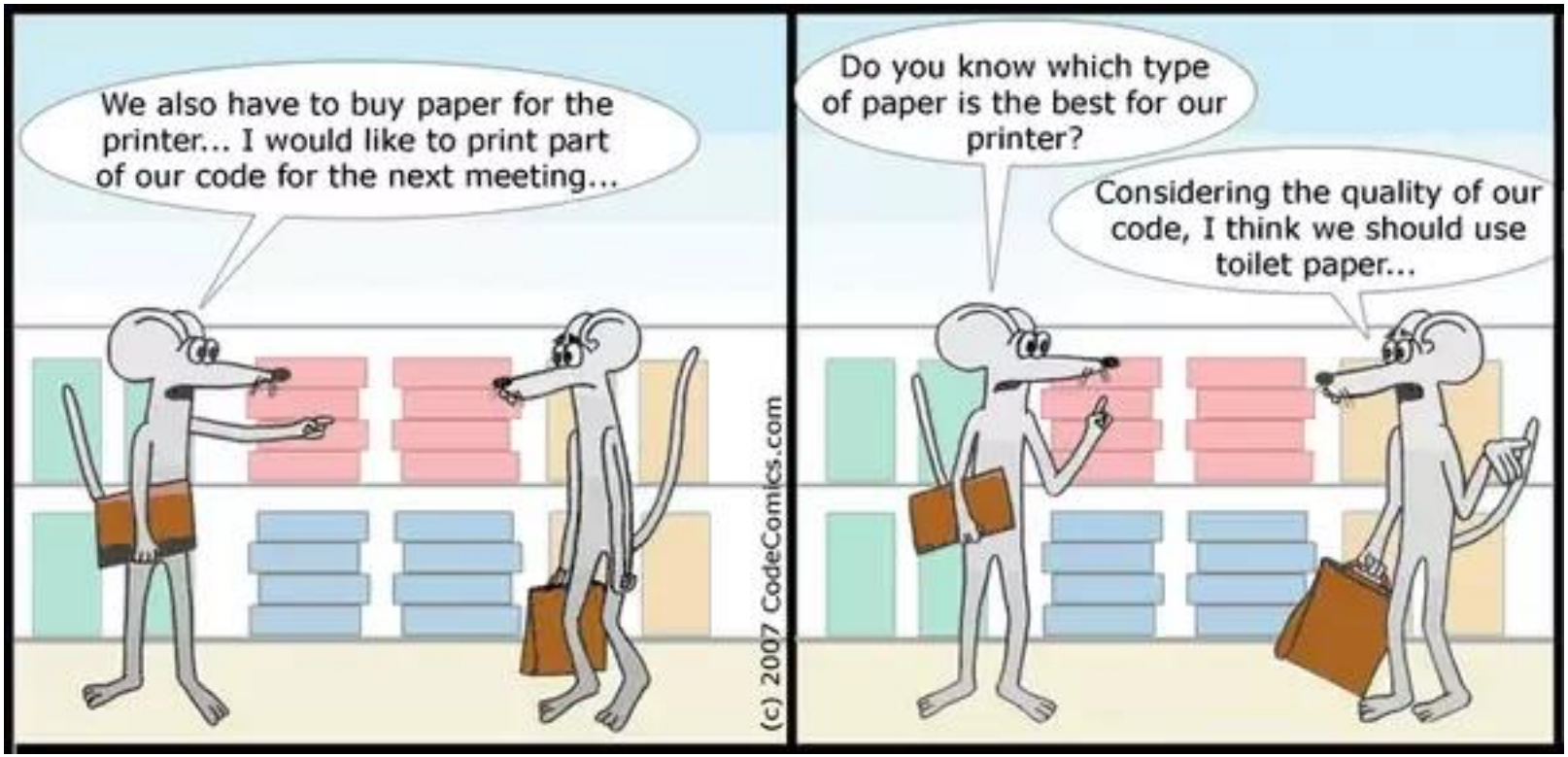
- Cadrul extins este utilizat atunci când:
  - **o excepție nesecurizată (*Non-secure code*) întrerupe o excepție securizată (*Secure code*)**
  - **sau în cazul utilizării excepțiilor cu sosire întârziată și excepția finală este securizată**

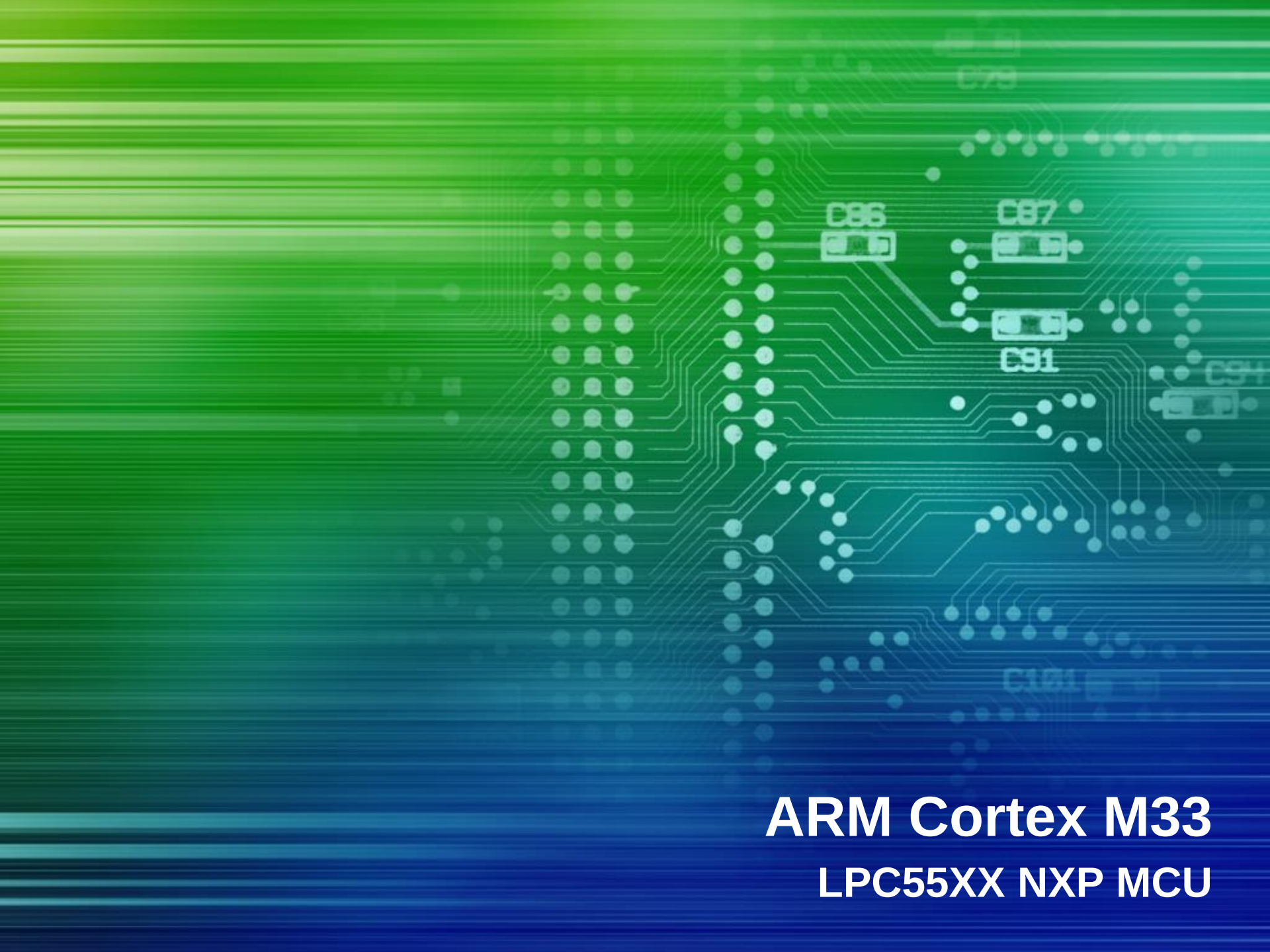
Stiva – Tipuri

- După cum am observat … există 4 principii de funcționare a stivei
- Arhitecturi diferite … tipuri de stivă diferite

Tip de stivă	Principiul de funcționare al stivei
Full Ascen.	Crescătoare, SP indică ultimul element introdus
Empty Ascen.	Crescătoare, SP indică primul loc liber
Full Descen.	Descrescătoare, SP indică ultimul element introdus
Empty Descen.	Descrescătoare, SP indică primul loc liber

Arhitectură	Principiul de funcționare al stivei
Intel 8086	Descrescătoare, SP indică ultimul element introdus
Intel 8051	Crescătoare, SP indică ultimul element introdus
ARM v5	CONFIGURABIL (oricare dintre cele 4 tipuri)
ARM v8-M	CONFIGURABIL (oricare dintre cele 4 tipuri)


We also have to buy paper for the printer... I would like to print part of our code for the next meeting...
(c) 2007 CodeComics.com
Do you know which type of paper is the best for our printer?
Considering the quality of our code, I think we should use toilet paper...

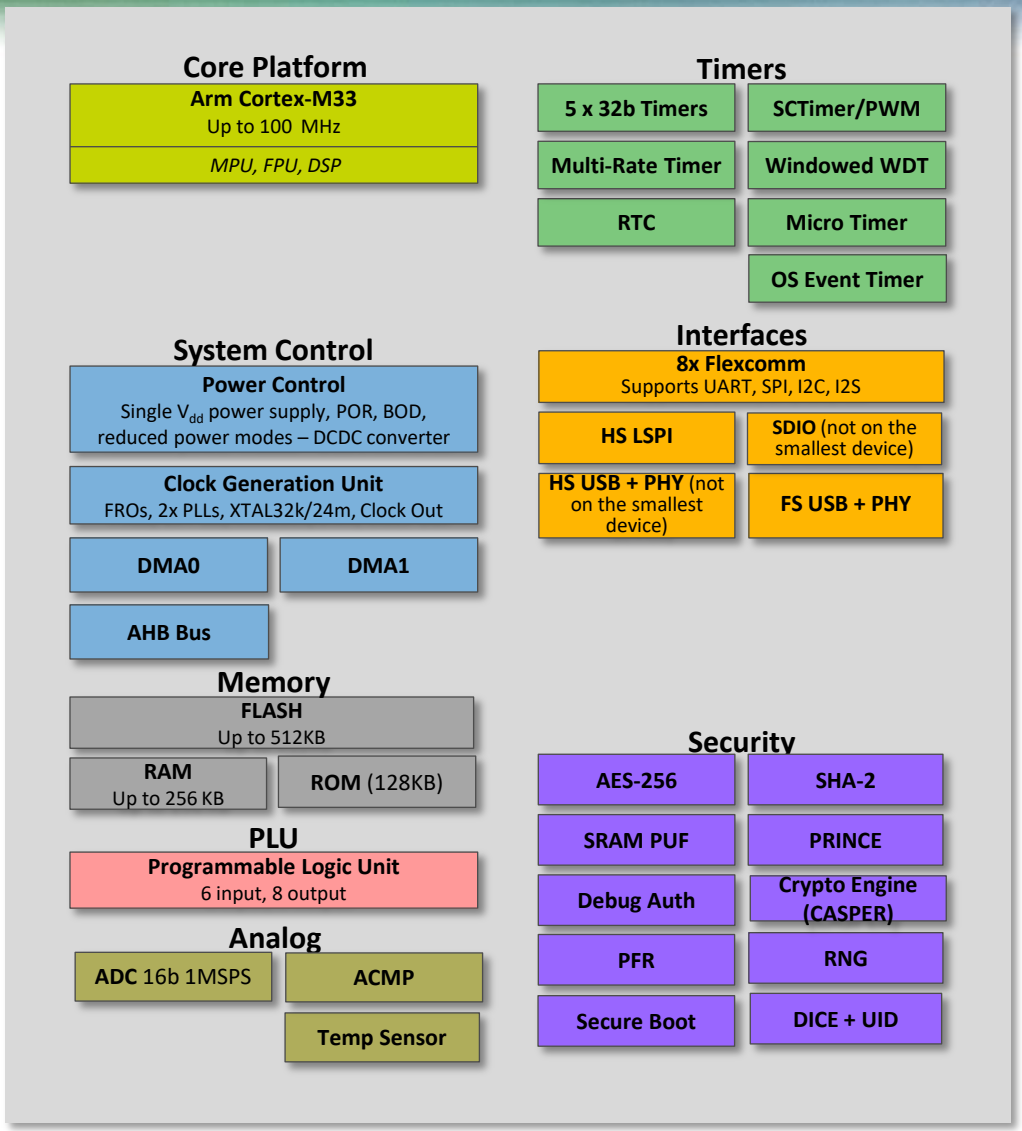


ARM Cortex M33

LPC55XX NXP MCU

NXP LPC55XX MCU

LPC55S2x / LPC552x



Core Platform

- Up to 100MHz Cortex-M33
  - Code Encryption
- Multilayer Bus Matrix

Advanced Security (LPC55S2x only)

- AES-256 HW Encryption/Decryption Engine
- SHA-2
- SRAM PUF for Key Generation support
- PRINCE Real-time encrypt/decrypt for flash data
- Asymmetric Crypto Engine (CASPER)
- Debug authentication
- RNG
- Protected Flash Region (PFR)
- Secure Boot

Analog

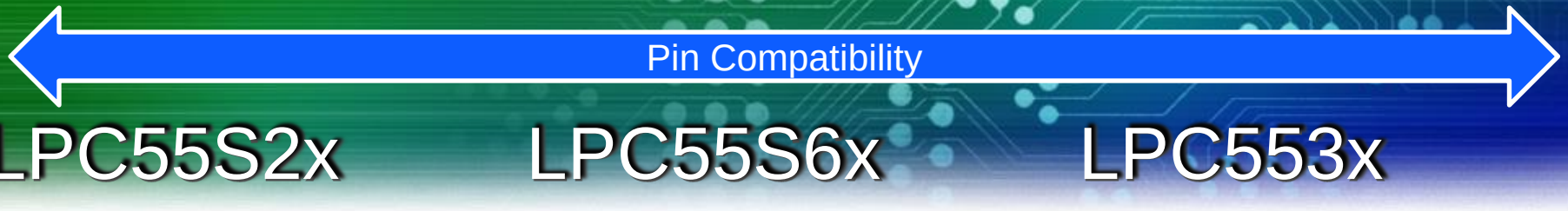
- 16b ADC, 10ch (5-diff pairs), 1MSPS
- Analog Comparator
- Temperature Sensor
- Crystal 24M

Packages

- **LQFP100, VFBGA98**
- QFP64

Other

- Buck DC-DC
- Operating voltage: 1.8 to 3.6V
- Temperature range: -40 to 105 °C



LPC55S2x

Core Platform

Arm Cortex-M33
Up to 100 MHz
MPU, FPU, DSP

System Control

Power Control
Single V_{dd} power supply, POR, BOD, reduced power modes – DCDC converter

Clock Generation Unit
FROs, 2x PLLs, XTAL32k/24m, Clock Out

DMA0 | **DMA1**

AHB Bus

Timers

5 x 32b Timers | **SCTimer / PWM**
Multi-Rate Timer | **Windowed WDT**
RTC | **Micro Timer**
OS Event Timer

Interfaces

8x Flexcomm
Supports UART, SPI, I2C, I2S

HS LSPI | **SDIO**
HS USB + PHY (not on the smallest device) | **FS USB + PHY**

Memory

FLASH
Up to 512KB

RAM
=< 256 KB | **ROM** (128KB)

Security

AES-256 | **SHA-2**
SRAM PUF | **PRINCE**
Debug Auth | **Crypto Engine**
PFR | **RNG**
Secure Boot | **DICE + UID**

PLU

Programmable Logic Unit
6 input, 8 output

Analog

ADC 16b 1MSPS | **ACMP** | **Temp Sensor**

LPC55S6x

Core Platform

Arm Cortex-M33
Up to 100 MHz
TrustZone, MPU, FPU, DSP

Arm Cortex-M33
Up to 100 MHz

DSP Accelerator

Timers

5 x 32b Timers | **SCTimer / PWM**
Multi-Rate Timer | **Windowed WDT**
RTC | **Micro Timer**
OS Event Timer

System Control

Power Control
Single V_{dd} power supply, POR, BOD, reduced power modes – DCDC converter

Clock Generation Unit
FROs, 2x PLLs, XTAL32k/24m, Clock Out

Secure DMA0 | **Secure DMA1**

Secure AHB Bus

Interfaces

8x Flexcomm
Supports UART, SPI, I2C, I2S

HS LSPI | **SDIO**
HS USB + PHY | **FS USB + PHY**

Security

AES-256 | **SHA-2**
SRAM PUF | **PRINCE**
Debug Auth | **Crypto Engine**
PFR | **RNG**
Secure Boot | **DICE + UID**

Memory

FLASH
Up to 640KB

RAM
=< 320 KB | **ROM** (128KB)

PLU

Programmable Logic Unit
6 input, 8 output

Analog

ADC 16b 1MSPS | **ACMP** | **Temp Sensor**

LPC553x

Core Platform

Arm Cortex-M33
Up to 100 MHz
TrustZone, MPU, FPU, DSP

System Control

Power Control
Single V_{dd} power supply, POR, BOD, reduced power modes – DCDC converter

Clock Generation Unit
FROs, 2x PLLs, XTAL32k/24m, Clock Out

Secure DMA0 | **Secure DMA1**

Secure AHB Bus

Timers

5 x 32b Timers | **SCTimer / PWM**
Multi-Rate Timer | **Windowed WDT**
RTC | **Micro Timer**
OS Event Timer

Interfaces

8x Flexcomm
Supports UART, SPI, I2C, I2S

HS LSPI | **CAN-FD**
HS USB + PHY | **FS USB + PHY**

Security

AES-256 | **SHA-2**
SRAM PUF | **PRINCE**
Debug Auth | **Crypto Engine**
PFR | **RNG**
Secure Boot | **DICE + UID**
Code WDG

Memory

FLASH
Up to 256KB

RAM
=< 96KB | **ROM** (128KB)

PLU

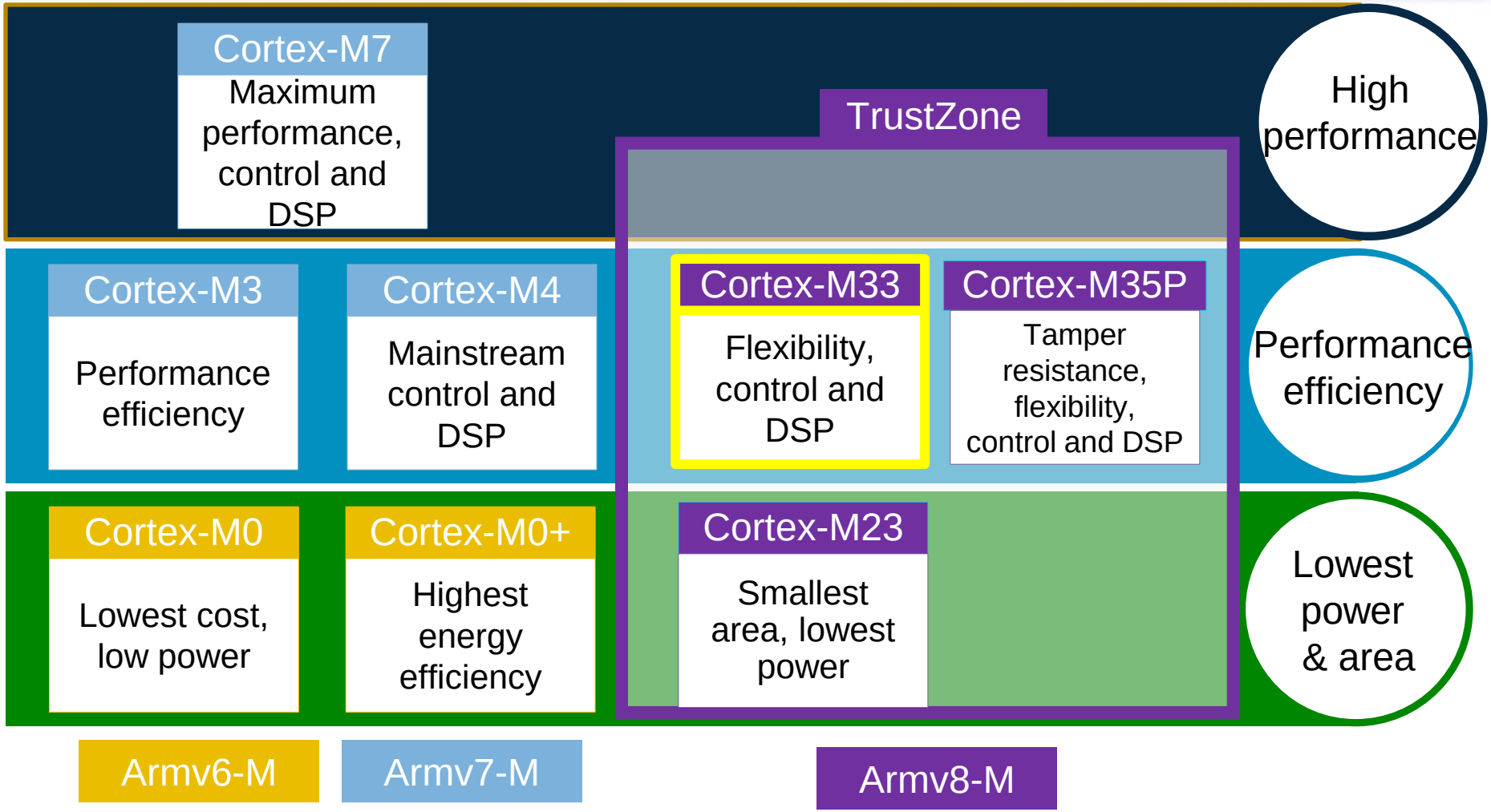
Programmable Logic Unit
6 input, 8 output

Analog

ADC 16b 1MSPS | **ACMP** | **Temp Sensor**

ARM Cortex-M33 Core

Cortex-M processor portofolio

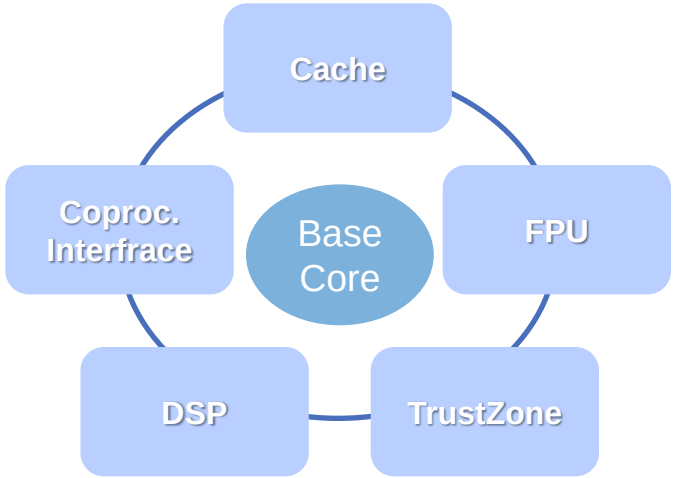


Efficiency, security and flexibility

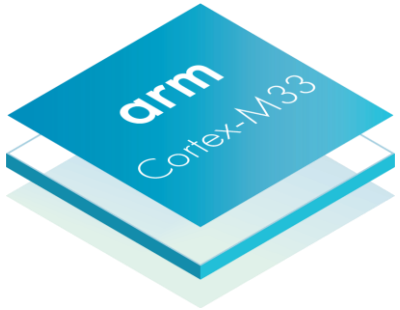
Extremely compact

Cortex-A5, Cortex-M33
size based on 40nm

Configurable and extensible



Widely applicable



Security for diverse embedded markets

32-bit processor of choice

- Optimal balance between performance and power
- 20% greater performance than Cortex-M4
- With TrustZone, same energy efficiency as Cortex-M4
- Binary compatible with Cortex-4 when TrustZone disabled

Digital signal control

- Bring DSP to all developers
- FPU offering up to 10x performance over software

Extensible compute

- Co-processor interface for tightly-coupled acceleration

Security foundation

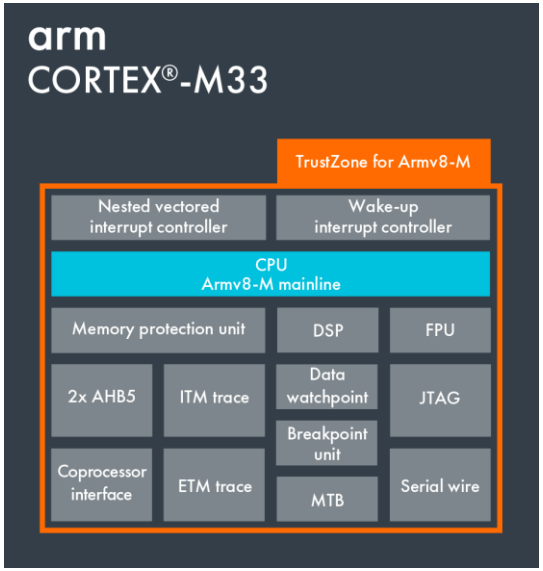
- System-wide security with TrustZone technology

Enhanced memory protection

- Easy to program
- Dedicated protection for both secure and non-secure states

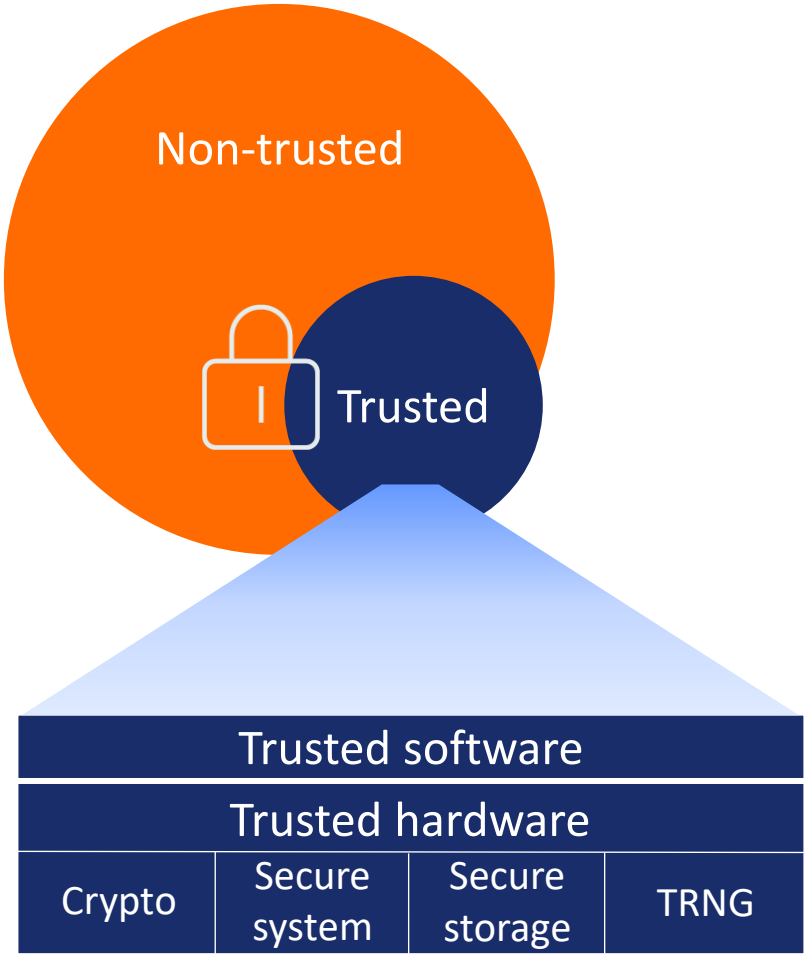
Enhanced & secure debug

- Security aware debug
- Simplified firmware development



TrustZone + Platform Security Architecture (PSA)

A comprehensive security foundation



Security separation with TrustZone

- Isolate trusted resources from non-trusted
- Reduce attack surface of key components

Security throughout the system

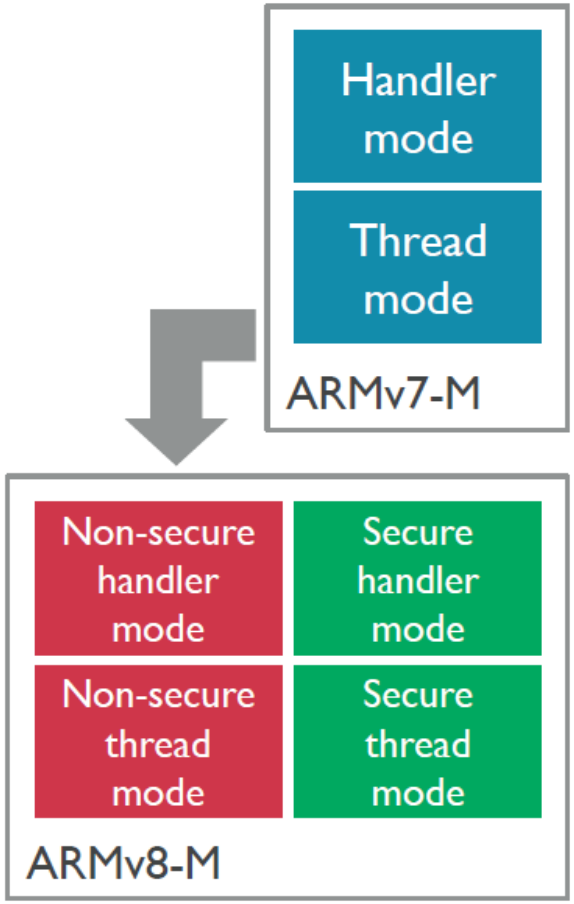
- Software, CPU, interconnect, memory and peripherals

Trusted hardware

- Fortified security for entire device lifecycle

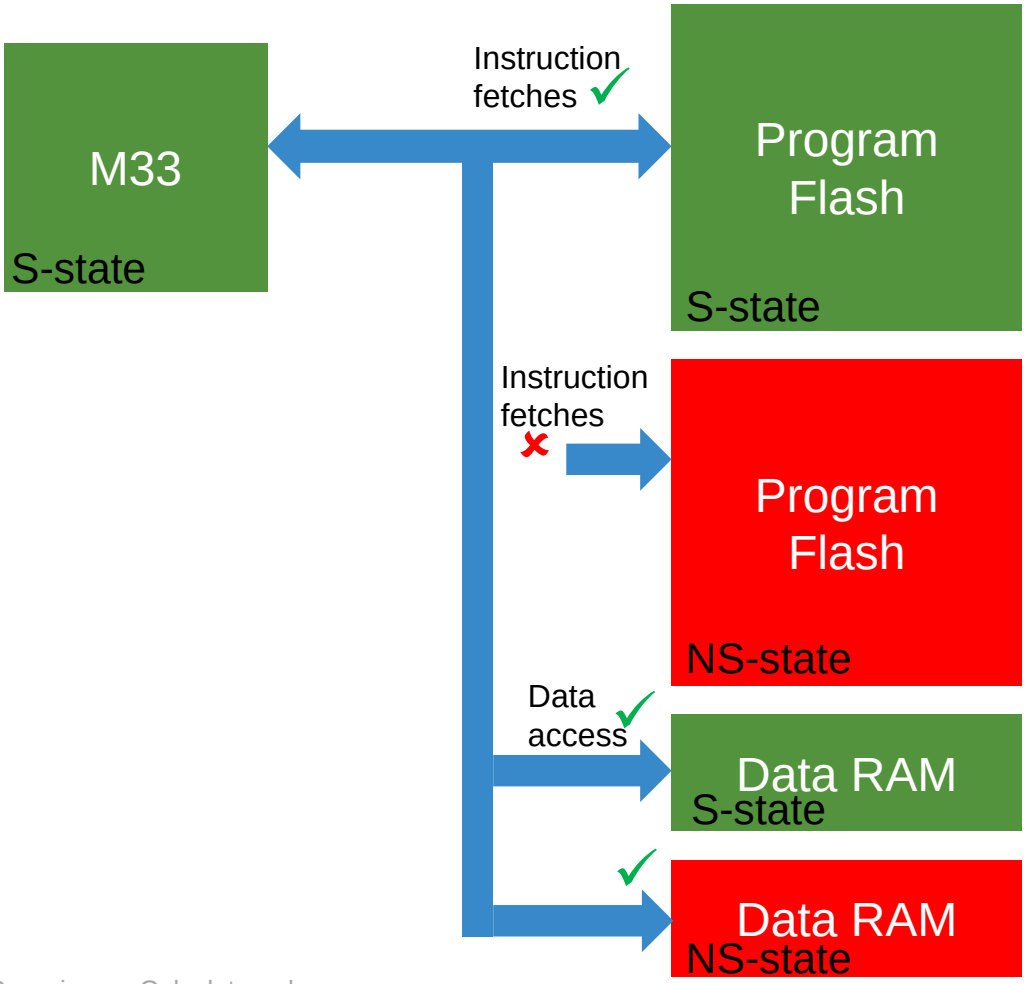
ARMv8-M additional states

- Secure and non-secure code runs on a single CPU
  - For efficient embedded implementation
- Secure state for trusted code
  - New secure stack pointers for robust operation
  - Addition of stack-limit checking
- Dedicated resources for isolation between domains
  - Separate memory protection units for secure and non-secure
  - Private SysTick timer for each state
- Secure side can configure target domain of interrupts



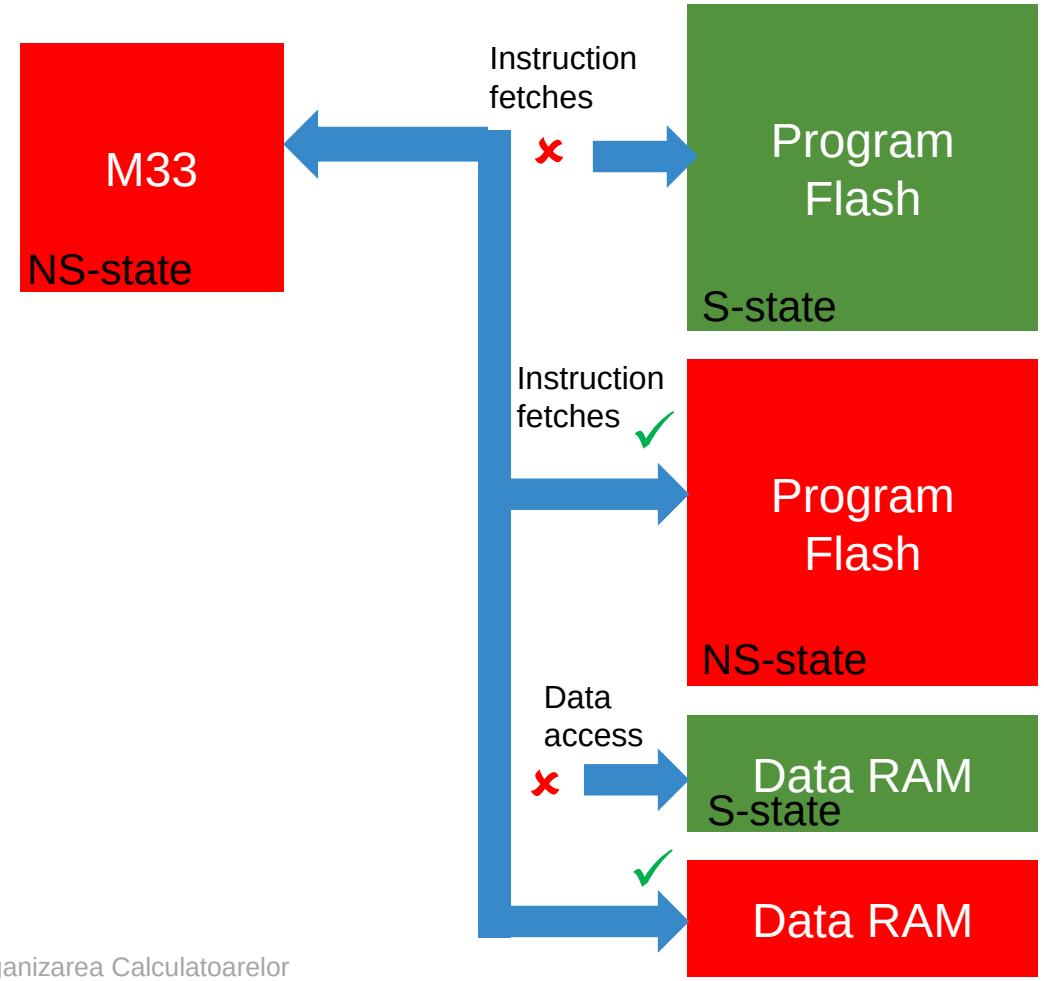
Secure CPU state

- CPU in secure state can only execute from Secure Program memory.
- CPU in secure state can access data from both secure and NS memory.



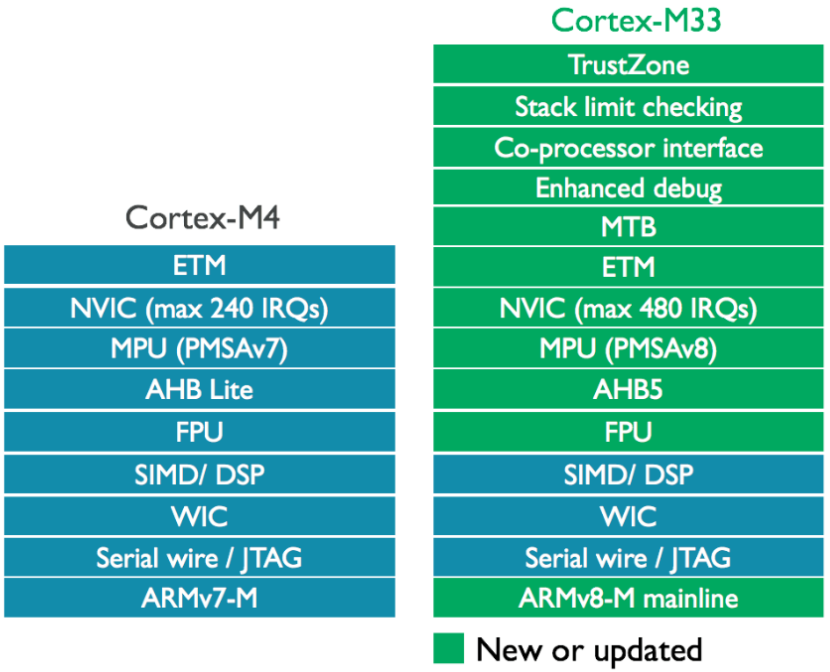
Non-Secure CPU state

- CPU in non-secure state can only execute from non-secure program memory.
- CPU in non-secure state can access data from both NS memory only.



Cortex M33 and Cortex M4

- 20% greater performance than Cortex-M4
- With TrustZone, same energy efficiency as Cortex-M4
- Key Features of Cortex-M33
  - TrustZone for ARMv8-M
  - Co-Processor Interface for extensibility
  - Memory Protection Unit (MPU) for task isolation
  - DSP Extension
  - Single Precision Floating Point Unit



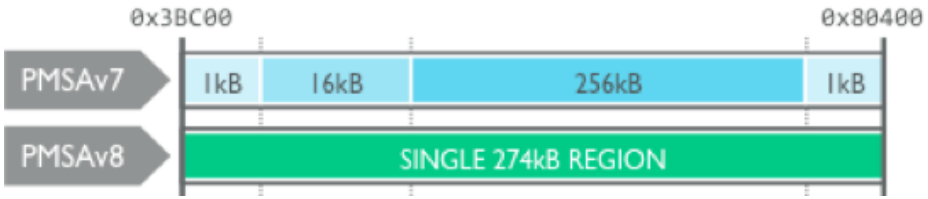
Performance of the Cortex-M Processors

	Dhrystone DMPIS/MHz (v2.1) – official	Coremark/MHz (v1.0)
Cortex-M3	1.25	3.32
Cortex-M4	1.25	3.40
Cortex-M33	**1.5**	**3.86**

- Cortex-M33 has a redesigned efficient pipeline which allows execution of up to two instruction in the same clock cycle
- Similar to previous Cortex-M3 and Cortex-M4 cores, but with much **better flexibility** in system design, **better energy efficiency** and **higher performance**

Memory Protection Unit – Cortex-M33

- Based on protected memory system architecture **PMSAv8**
- Provides 16 regions for each of the secure and non-secure states
- Adopts base and limit style comparators for regions
  - Each region has a base starting address, ending address, and settings for access permission and memory attribute



Benefits

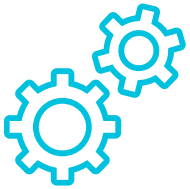
- Simplifies software development
- Encourages usage and reduces programming steps → Reduces context switch times
- OS can reprogram the MPU during task context switching to define memory access permissions for each task

Cortex-M33 Main Highlights

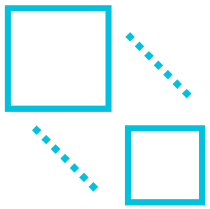
Architecture	Armv8-M Mainline (Harvard)
TrustZone	Optional TrustZone for Armv8-M
DSP Extensions	Optional DSP/SIMD instructions, Single cycle 16/32-bit MAC, Single cycle dual 16-bit MAC, 8/16-bit SIMD arithmetic
Floating Point Unit	Optional single precision floating point unit IEEE 754 compliant
Co-processor interface	Optional dedicated co-processor bus interface for up to 8 co-processor units for custom compute
Memory Protection	Optional Memory Protection Unit (MPU) with up to 16 regions per security state
Interrupts	Non-maskable Interrupt (NMI) and up to 480 physical interrupts with 8 to 256 priority levels
Wake-up Interrupt Controller	Optional for waking up the processor from state retention power gating or when all clocks are stopped
Debug	Optional JTAG and Serial Wire Debug ports. Up to 8 Breakpoints and 4 Watchpoints.
Trace	Optional Instruction Trace (ETM), Micro Trace Buffer (MTB), Data Trace (DWT), and Instrumentation Trace (ITM).

Cortex-M33 – Secure, feature rich IoT nodes

TrustZone for Armv8-M



Ease of Use



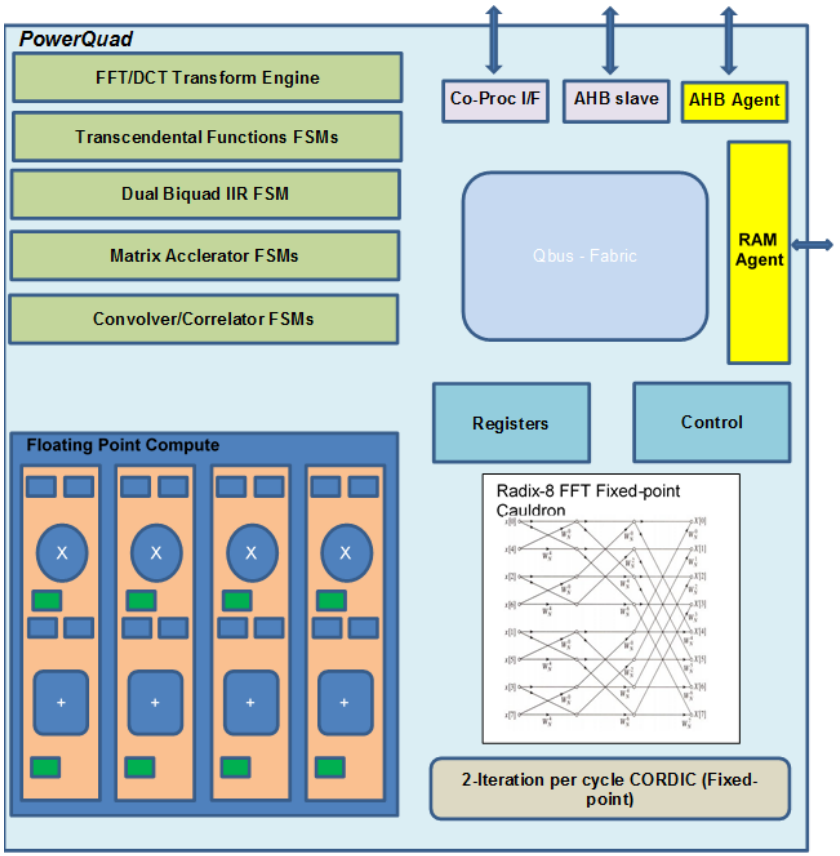
Scalability

- Capture part of the $30B IoT device market
  - By addressing the demand for a security standard for developers
  - By rapidly building a cost effective, diverse and easy-to-use product line
- Using the next-generation industry standard for secure embedded devices
  - Leverage widest developer base with minimal investment
  - Create a unique product by selecting the numerous configurations available at design and manufacturing
- Efficient, general purpose Armv8-M 32bit processor with Arm TrustZone
  - Optimal balance between area, power, security and performance
  - Signal processing, floating point and coprocessor interface

Co-Processor Accelerator PowerQuad

PowerQuad Hardware Accelerator DSP

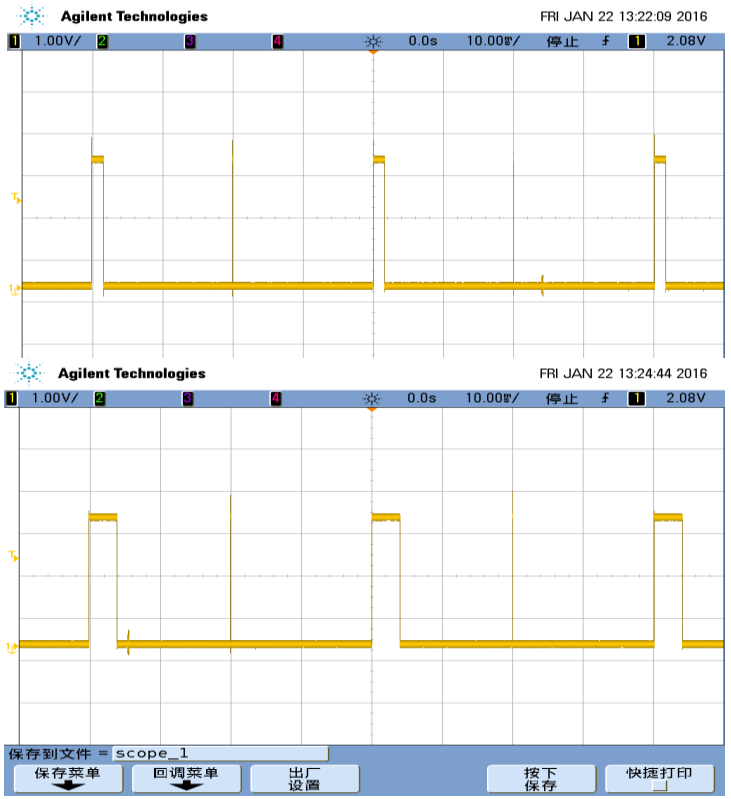
- Hardware Accelerator for Fixed and Floating Point Unit DSP Functions
- 5-10x Faster than Cortex-M33 for Matrix Operations, FIR/Convolution/Correlation
- ~15x Faster than Cortex-M33 running CMSIS-DIP lib for FFT/IFFT (~50x faster than generic FFT C-code running on Cortex-M33)
- **Key Advantages**
  - Energy consumption reduction for DSP tasks
  - Fast implementation for ‘lego’ blocks of DSP processing
  - Can Operate in parallel with Arm Cortex-M33 Core
  - No need for wasting RAM lookup tables



Why PowerQuad Co-Processor?

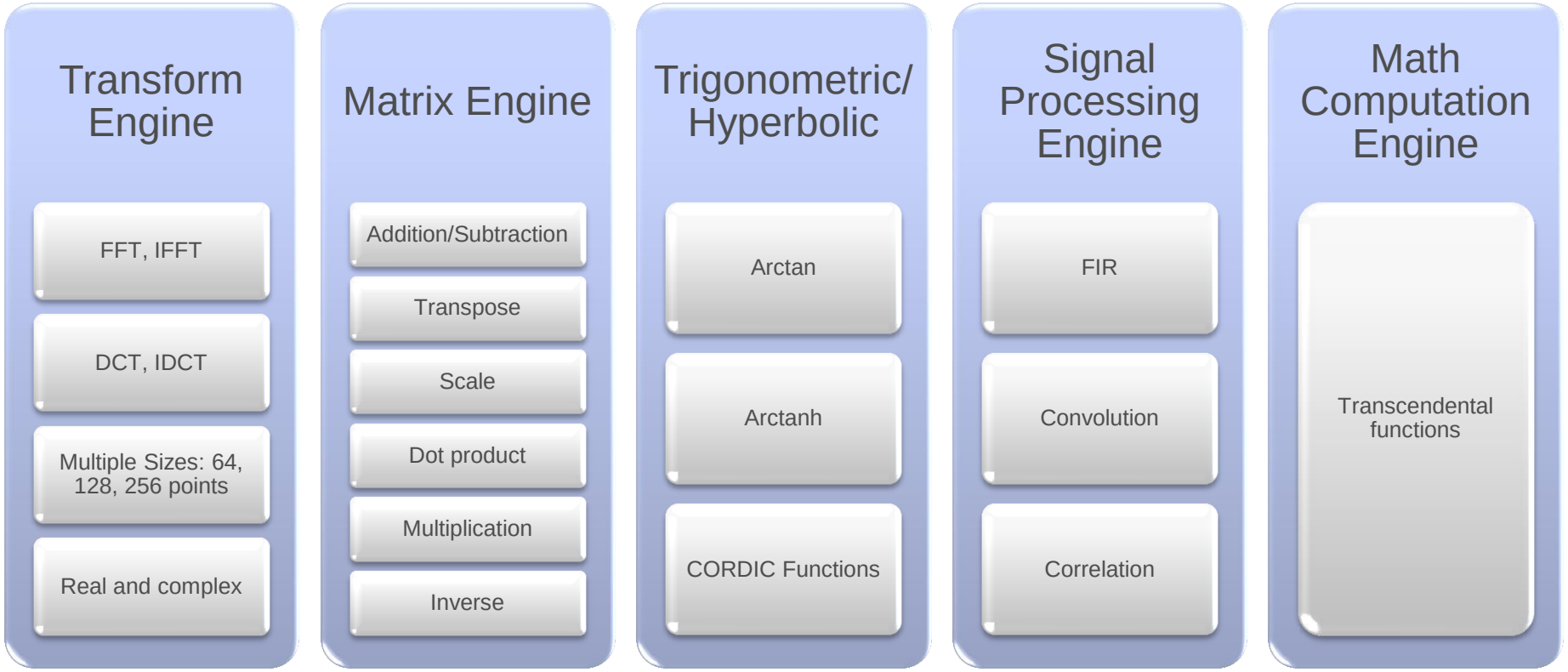
- PowerQuad is a hardware accelerator co-processor used for complex math calculations
  - Contains 4 x FP MAC
  - AHB Master allows PowerQuad to read/write data for input/computations/results
  - PowerQuad can execute to 2 co-proc instructions in parallel

	Plain C FXP with -Ofast opt on Cortex M33	CMSIS DSP	PowerQuad
FFT real N=32	6163	2295	273
FFT real inverse N=32	8951	2453	329
FFT real N=64	13856	5718	465
FFT real inverse N=64	20517	6009	553
FFT real N=128	30761	10798	1066
FFT real inverse N=128	46255	11394	1235
FFT real N=256	67846	26333	2078
FFT real inverse N=256	103181	27541	2353
FFT real N=512	148283	50574	4062
FFT real inverse N=512	227651	53191	4595



LPC5500 PowerQuad – Mapped Functions

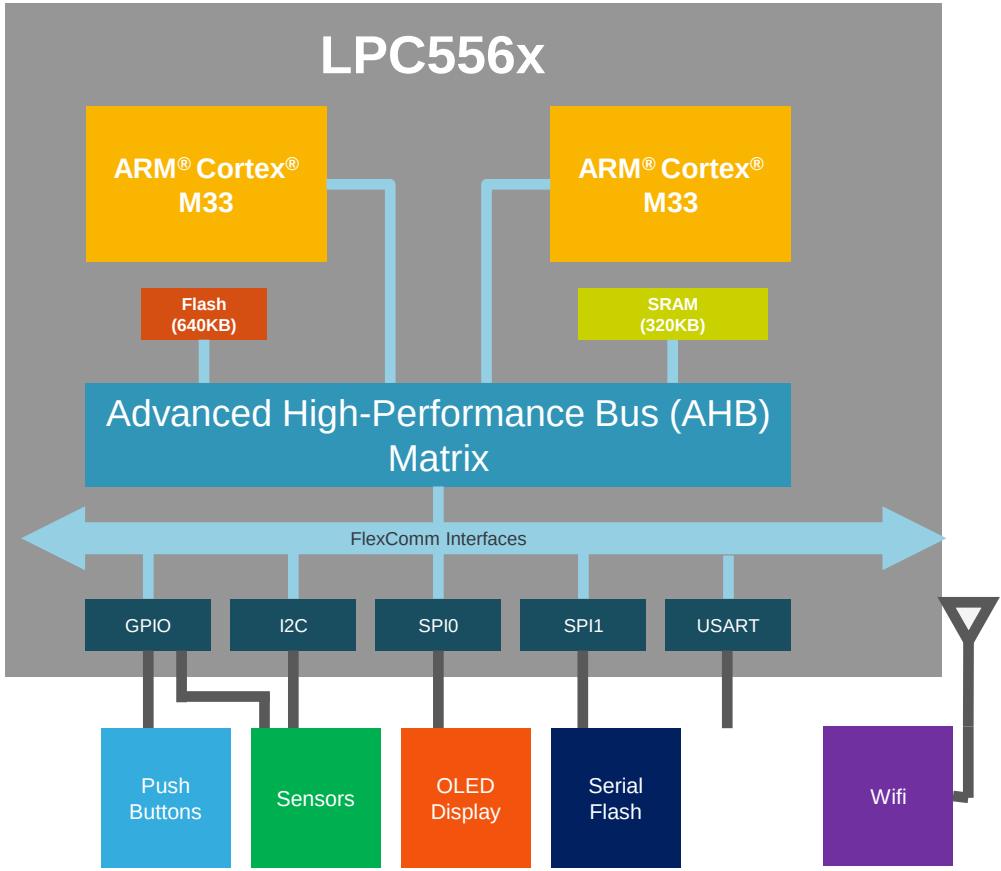
- HW accelerator for frequently used math and signal processing computations
  - Ex: Motion context, Voice Recognition, Neural networks



LPC55XX
System Infrastructure
Bus, DMA, Clock, Power Management

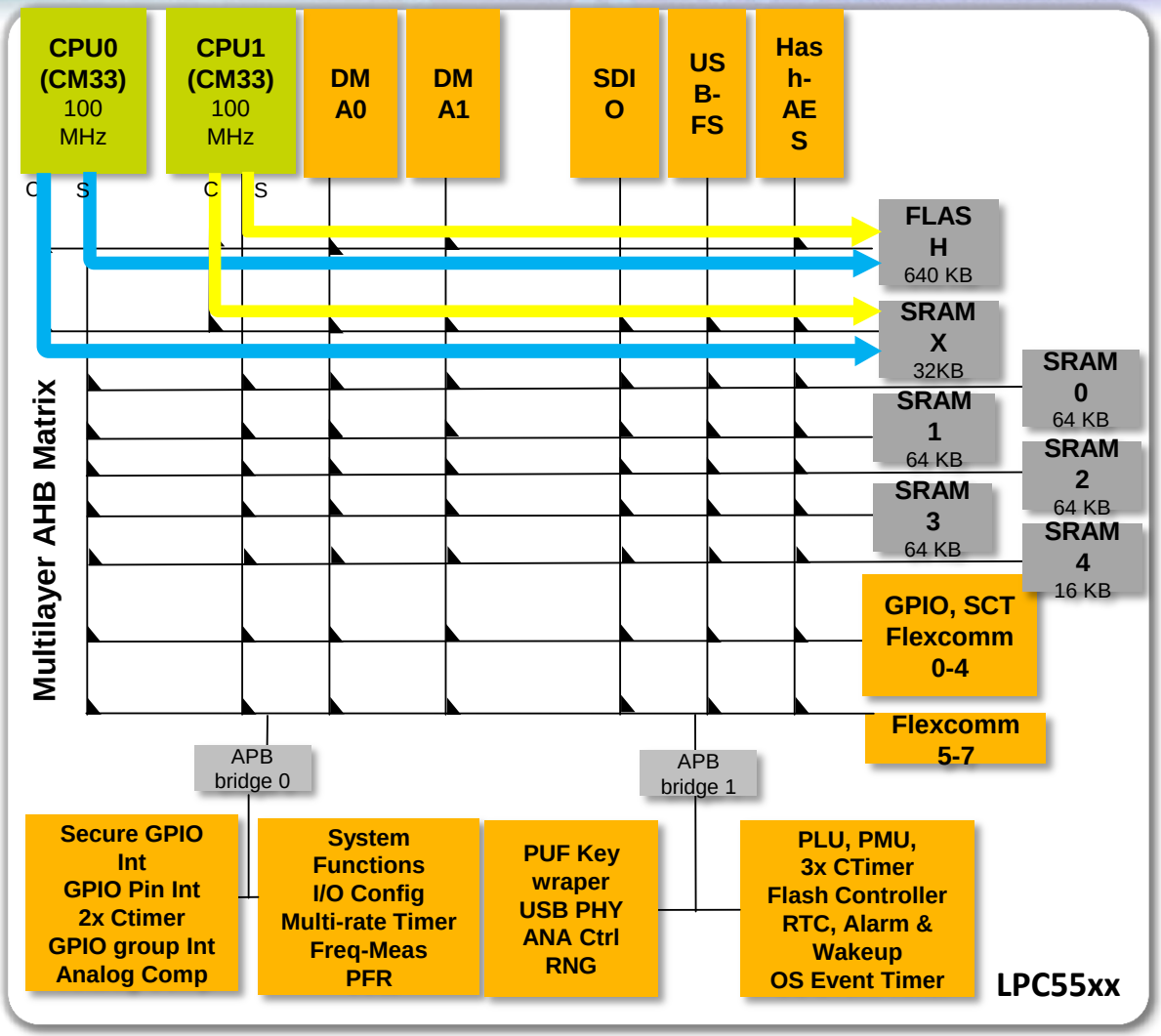
Asymmetric dual-core MCU architecture

- **Dual Core**
  - CPU0
    - Cortex-M33 with NVIC, FPU, MPU, DSP, ITM, SAU, TrustZone Security extension
    - Configured as primary CPU: Boots on Reset de-assertion, cannot be disabled
  - CPU1
    - Cortex-M33 with NVIC
    - Stays in reset until enabled by CPU0
  - Application partitioning
    - CPU1 can be leveraged for running system level tasks and the CPU0 can wake up to process data intensive tasks leveraging the co-processors
  - Ease of software development
    - Software development teams can develop code for each core independently allowing for a faster time to market
- **Low Power Always-On Operation**
  - Provides low power always-on listening from analog and digital interfaces



Bus Matrix configured for performance

- **Efficient Partitioning**
  - Allows both cores and DMAs to run at full speed with no contention
  - Memories and Peripherals are distributed on multiple slave ports and APB bridges allowing independently access from all bus masters
  - Contiguous system RAM
- **Programable Master priority**
- **Synchronous system clock**
  - Synchronous clock for all bus masters and slave AHB and APB register interfaces
  - Independent asynchronous peripheral interface clocks
  - Independent asynchronous timer/counter clocks



DMA Controller

DMA Channel#	DMA0	DMA1
0	Hash-AES	Hash-AES
1	Trigger Events/SW	Trigger Events/SW
2-3	HS-SPI	HS-SPI
4-7	FC0-1	FC0-1
8-9	FC3	FC3
8-11	FC2	
12-19	FC4-FC7	
20	Trigger Events/SW	
21-22	ADC	

DMA Trigger sources		
GPIO INT0-3	5x Timer Match0/1	Comparator
SCT	AES	

- **DMA Channels**
  - **DMA0:** 23 channels
  - **DMA1:** 10 channels
  - User programmable channel priority
  - Continuous priority arbitration
  - Supports single transfers up to 4kB
  - Both DMAs can operate in parallel
  - Either DMA can be configured as Secure or Non-Secure DMA in Trusted execution application

- **DMA Triggers:**
  - **DMA0:** 22-inputs
  - **DMA1:** 15 inputs
  - Various on-chip and off-chip trigger sources
  - Any channel can be mapped to DMA trigger based transfers
  - Can be used to initiate Memory to Memory
  - Triggers can be chained

Clock Sources

XTAL32K
Source for RTC, OS-Timer. FCs

XTAL24M
Source for PLLs, HS-USB PHY, Bus-Clk, SCT

CLKIN
(XTAL Bypass mode)
Allows up to 80MHz clock input

FRO1M
15% accuracy
Source PMU, PLLs, WWDT, Ctimer, Bus-Clk, Flexcomms

FRO32K
+- 2% accuracy
Source for RTC, OS-Timer

PLL0
Spread-sprectrum/Fractional Wraper
Source for Bus-Clk, USBs , ADC Flexcomms SCT, SDIO

FRO96M
+-2% accuracy
Source Bus-Clk, Ctimer, ADC, USB-FS, Flexcomms, , SDIO

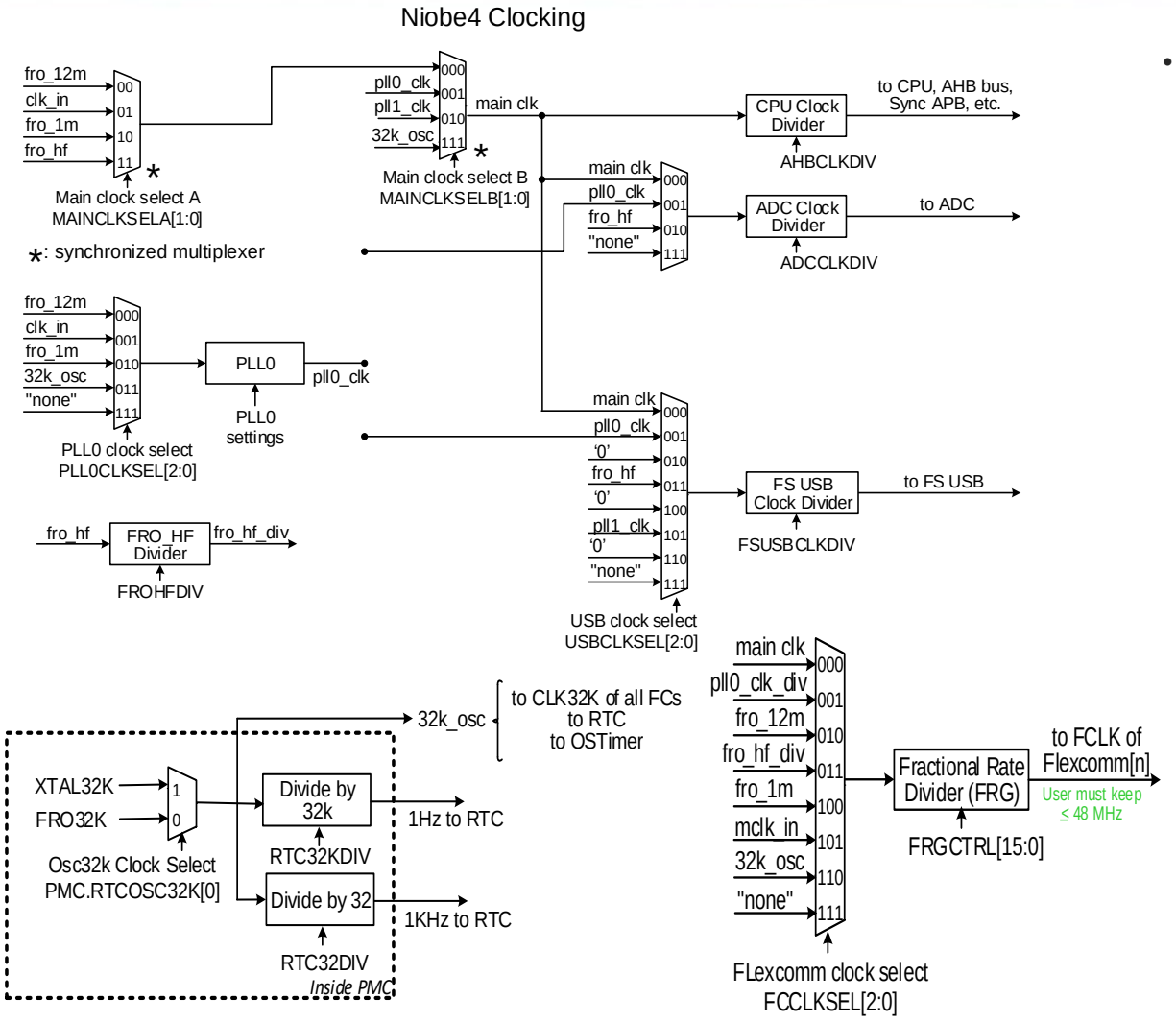
PLL0=System PLL, Audio PLL
PLL1=System PLL

USB PLL is not needed as HS-PHY can run on XTAL24M directly *(12M, 16M, 19.2M or 24M)*

PLL1
Source for Bus-Clk, USBs Flexcomms, , SDIO

FRO12M
+- 2% accuracy
Source Bus-Clk, PLLs, Ctimer, Flexcomms

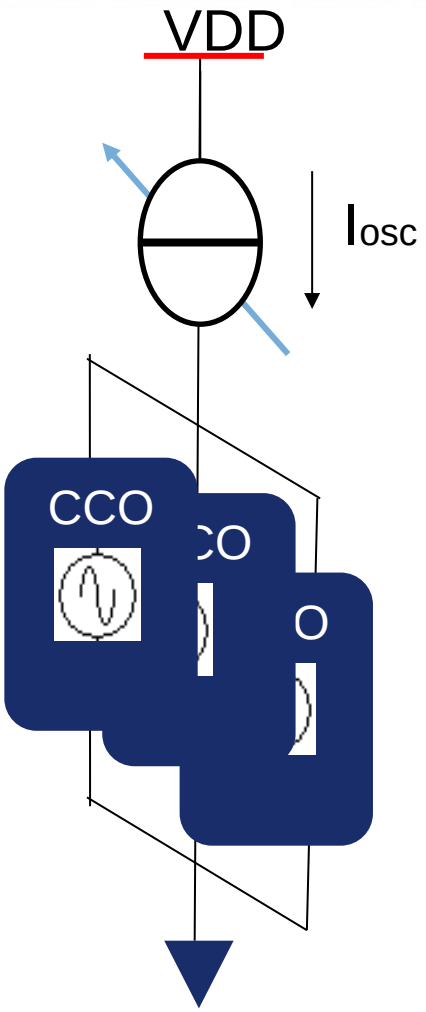
Clock Control



- **Clock Mux and Control**
  - Programable clock source selection
  - Synchronous mux for system-clock to allow dynamic clock-switching
  - System clock (main_clk) routable to all peripherals
  - Individual programmable clock-gate control to enable clocks to peripheral ports
  - All interface clocks are turned-off by default
  - Clockout mux and output pin

Free Running Oscillator (FRO)

- Low power internal Free-Running Oscillator
- Factory Trimmed 96 MHz/12 MHz
  - +/- 1% accuracy over entire voltage
- Some peripherals allow asynchronous operation from FRO while CPU operates from main clock
- FRO can be used as Main clock or PLL clock source
- Reduces dependency on System PLL
  - Benefit: fast restart after halting the CPU by sleep modes
  - Benefit: low power!



LPC55XX
Memory

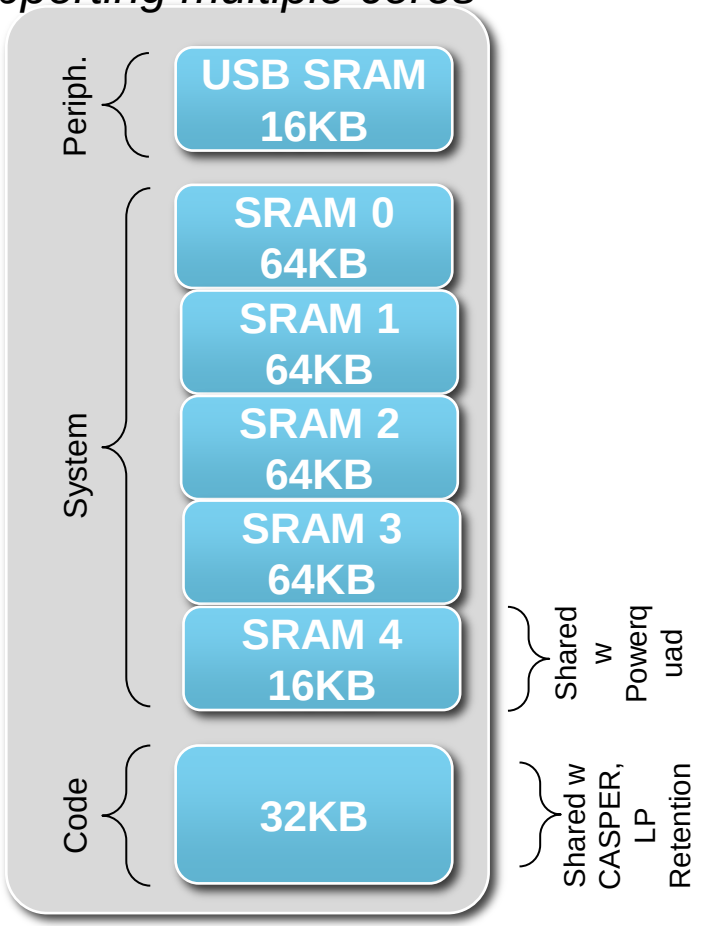
Flash Memory

- On-chip Memory
  - **Up to 640 KB on-chip Flash program memory**
    - 128 bit interface
    - 512 Byte page size
    - Multi page erase and single page program support
    - 1-bit ECC per 16 Byte Flash word
  - **8kB PFR area to support configurations (taken from 640kB)**
  - **Flash accelerator (FMC) with cache and prefetch**
    - 128 Byte buffer
  - **Supports real-time encryption and decryption using PRINCE**

SRAM

- Up to 320 KB total SRAM
  - **288 KB on System Bus**
    - 272 KB is contiguous in system space
    - 16kB USB RAM can be used by system when not used by USB-HS
  - **32 KB on Code Bus**
- Separate bus master access for higher throughput
- Individual power control for low-power operation

Segmented SRAM blocks for supporting multiple cores

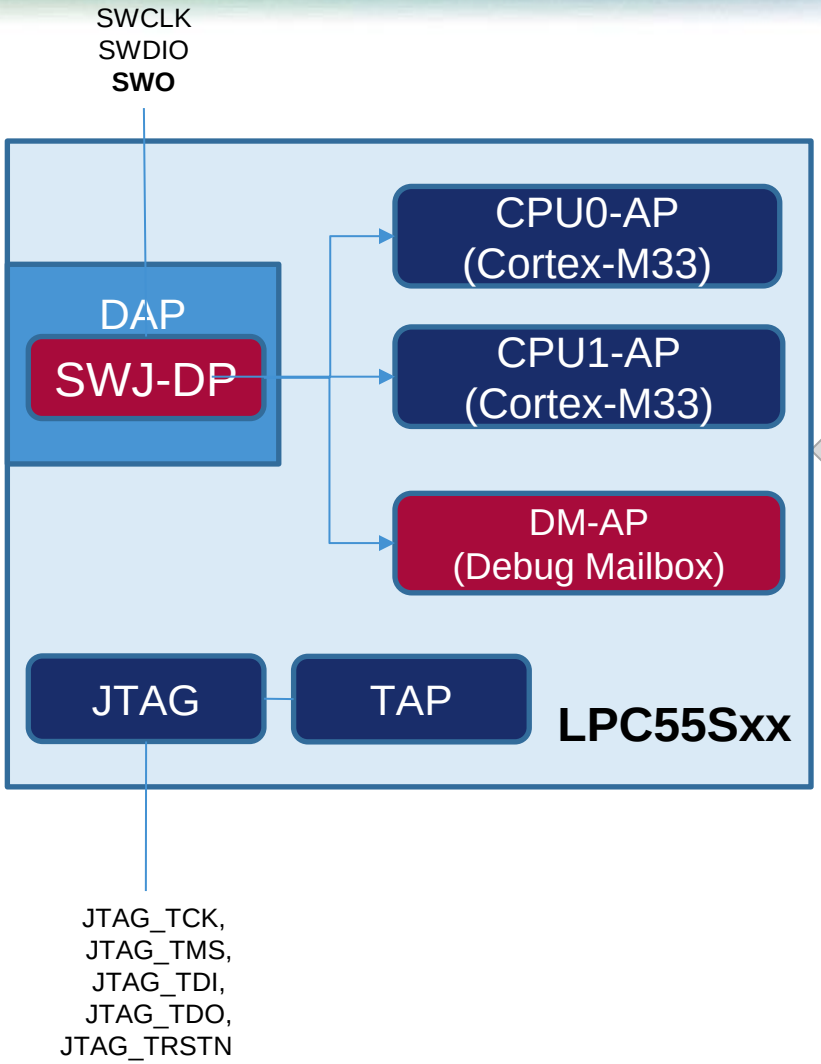


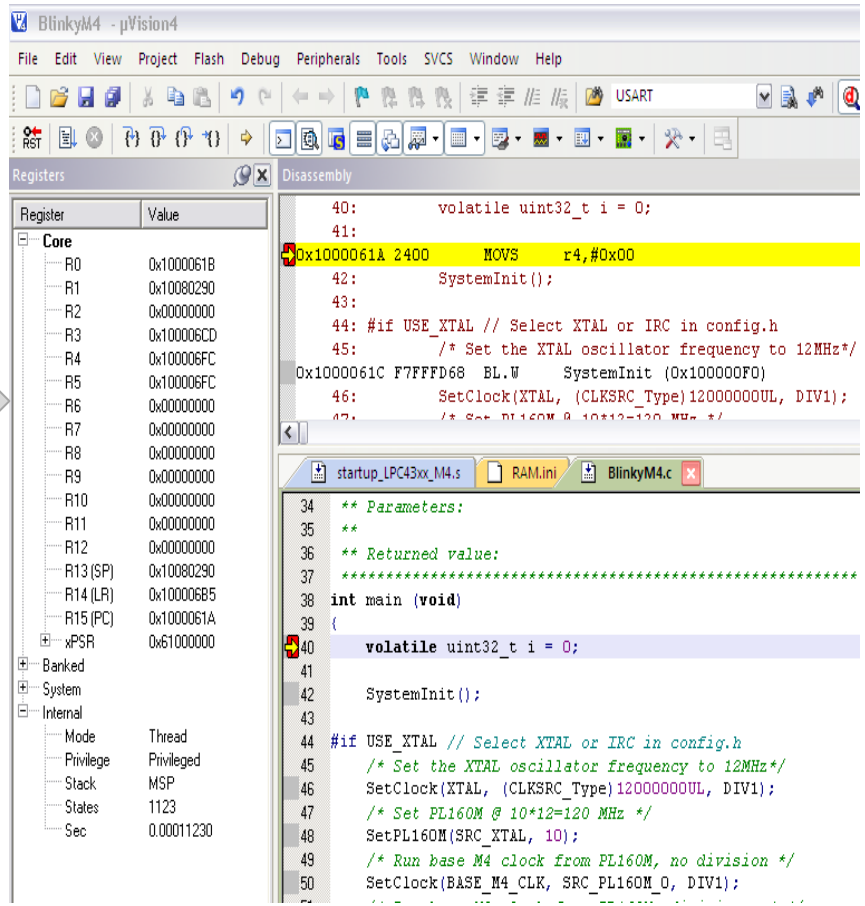
ROM

- 128KB on-chip ROM bootloader supporting:
  - Booting images from on-chip flash
  - CRC32 Support for image integrity checking
  - Flash In-Application Programming and In-System Programming (ISP)
  - Flash programming through ISP commands over following interfaces:
    - USB interface using HID Class device
    - UART interface with auto baud
    - SPI slave interface
    - I2C slave interface
  - ROM API functions

LPC55XX
Debug

Dual-Core Debug for the LPC5500





Debug Authentication Flow

1 Start Debug Mailbox Exchange

2 SYS_RESET_REQ

3 Debug Authentication Start (DBG_AUTH_START)

- Create DAC based on DCFG
- Generate 16bytes of random challenge vector

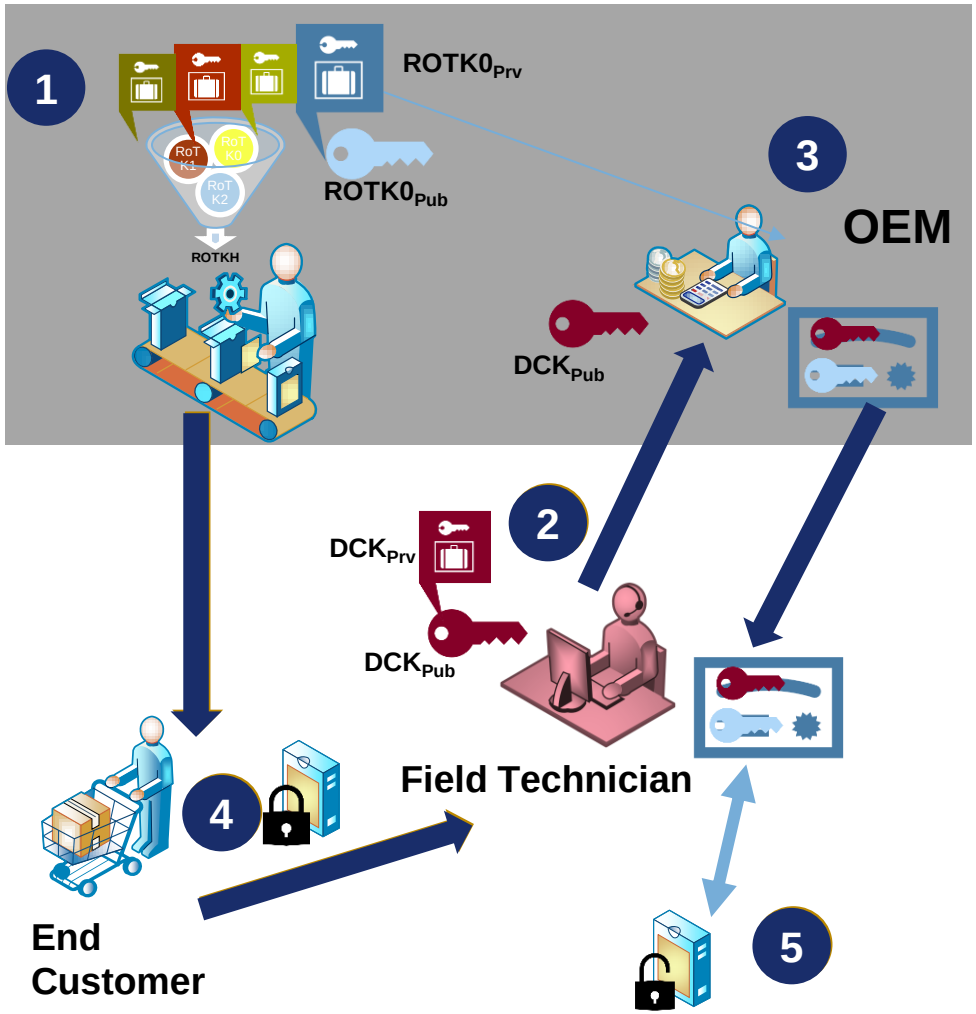
4 Debug Authentication Challenge (DAC)

- Find matching DC
- Encrypt Challenge vector
- Create DAR

5 Debug Authentication Response (DAR)

- Validate DC
- Validate DAR challenge data
- Opens debug access per credential

Debug Authentication Usage Example



1. OEM generates RoT key pairs and programs the device before shipping.
   - SHA256 hash of RoT public key hashes
2. Field Technician generates his own key pair and provides public key to OEM for authorization.
3. OEM attests the Field Technician's public key. In the debug credential certificate he assigns the access rights.
4. End customer having issues with a locked product takes it to Field technician.
5. Field technician uses his credentials to authenticate with device and un-locks the product for debugging.

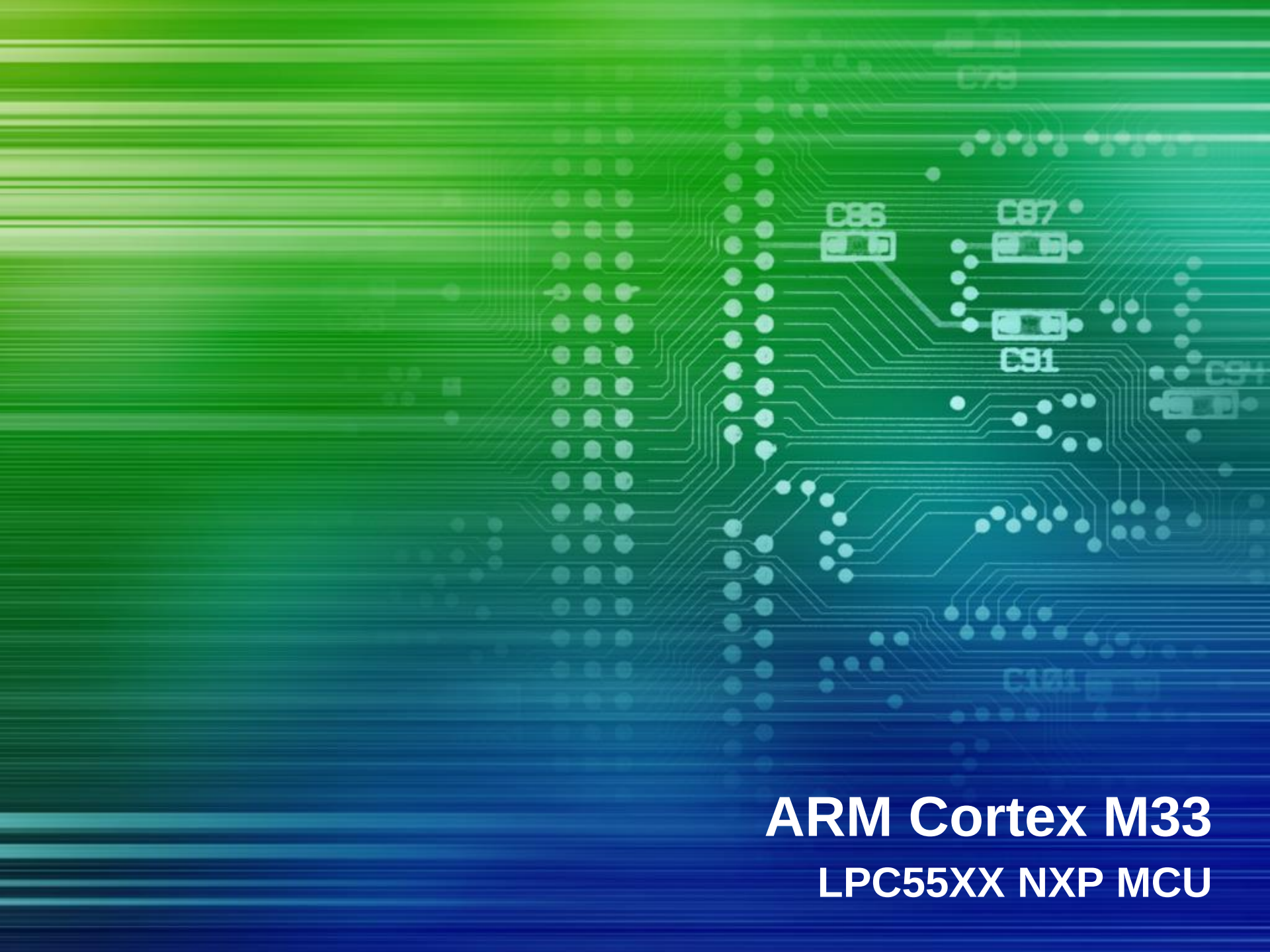
Debug Authentication Levels

Level	Features	System Requirement	Supported families
Extended	OEM can enable Tier-based debug access and distinguish trusted vs non-trusted debug access. Debug access to secure code/peripheral can be restricted to field technician with OEM authorization. Non-secure code can be accessible by any field-technician	TrustZone Subsystem (TZ, Secure Bus) Security subsystem (Hash, AES,PUF CASPER, Secure Boot) DM-AP, PFR	LPC55S6x LPC55S3x
Full	OEM can configure to disable debug access, and allow access to the field technicians with correct debug credentials	Security subsystem (Hash, AES, CASPER, Secure Boot) DM-AP, PFR	LPC55S2x LPC55S0x
Basic	OEM can configure to disable debug access using PFR setting, or leave it enabled	DM-AP, PFR	LPC553x LPC552x LPC551x

Referințe

https://www.arm.com/products/silicon-ip-cpu/cortex-m/cortex-m33

https://www.nxp.com/



ARM Cortex M33

LPC55XX NXP MCU

Power Management

Power Domains

Low Power Modes

Power Supplies

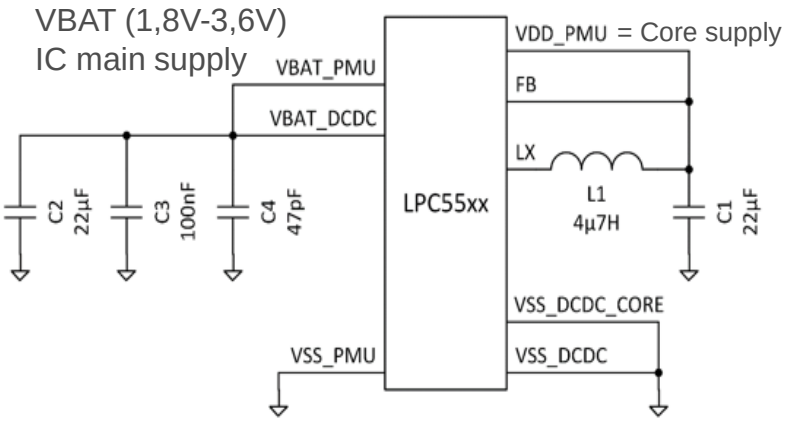
- **Power inside the part is supplied via 6 on-chip regulators:**
  - **DCDC** Bulk Converter [0.950V – 1.200V], supplies main digital core logic
  - **LDO_DEEP_SLEEP** regulator [0.9V – 1.075V], used during “Deep-sleep”
  - **LDO_AO** regulator [0.7V – 1.220V], supplies the “Always on” digital logic
  - **LDO_MEM** regulator [0.7V – 1.220V], supplies SRAM during low power modes
  - **LDO_USB_HS** regulator [1.8V – 2.0V], supplies USB High Speed Interface
  - **LDO_FLASH_NV** regulator [1.8V – 2.0V], supplies Flash Macro

- **All 6 previously mentioned internal regulators are supplied by the main external supply called VBAT [1.8V – 3.6V]**

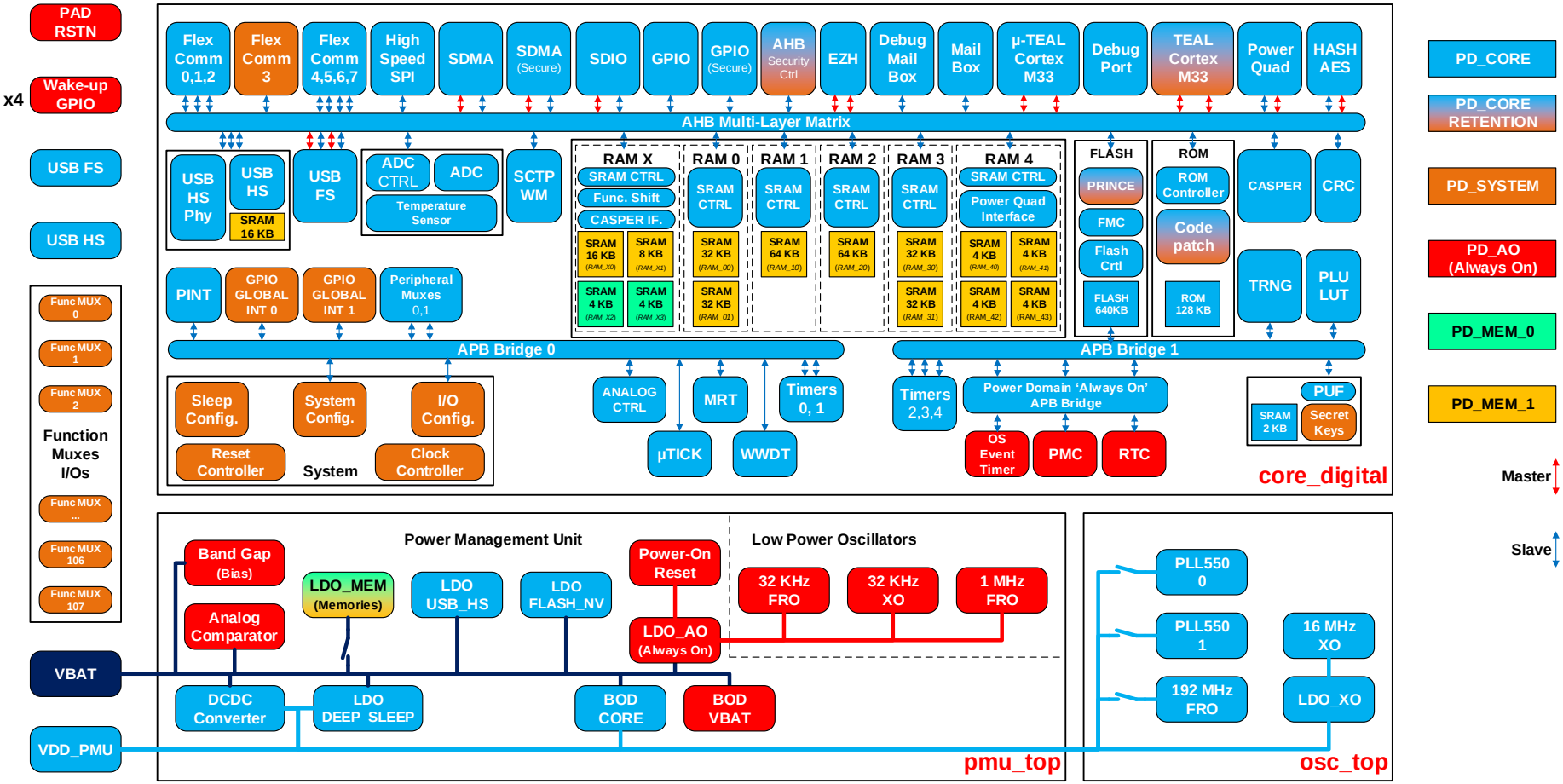
Power Management – DC-DC Converter

Up to +80% Efficiency DC-DC that is directly controlled and started by the Power Management Controller (PMC)

- **Features**
  - Input Voltage range: 1.8V to 3.6V
  - Output voltage: 1.1V
  - Max Load current: 50mA
  - Unconditional stable, Constant ON-Time PFM (COT)
  - Low Ripple with proper user COT trimming and optimal valued external inductor and capacitor
  - **DC-DC needs a 4µ7H inductor and a 22µF ceramic capacitor**



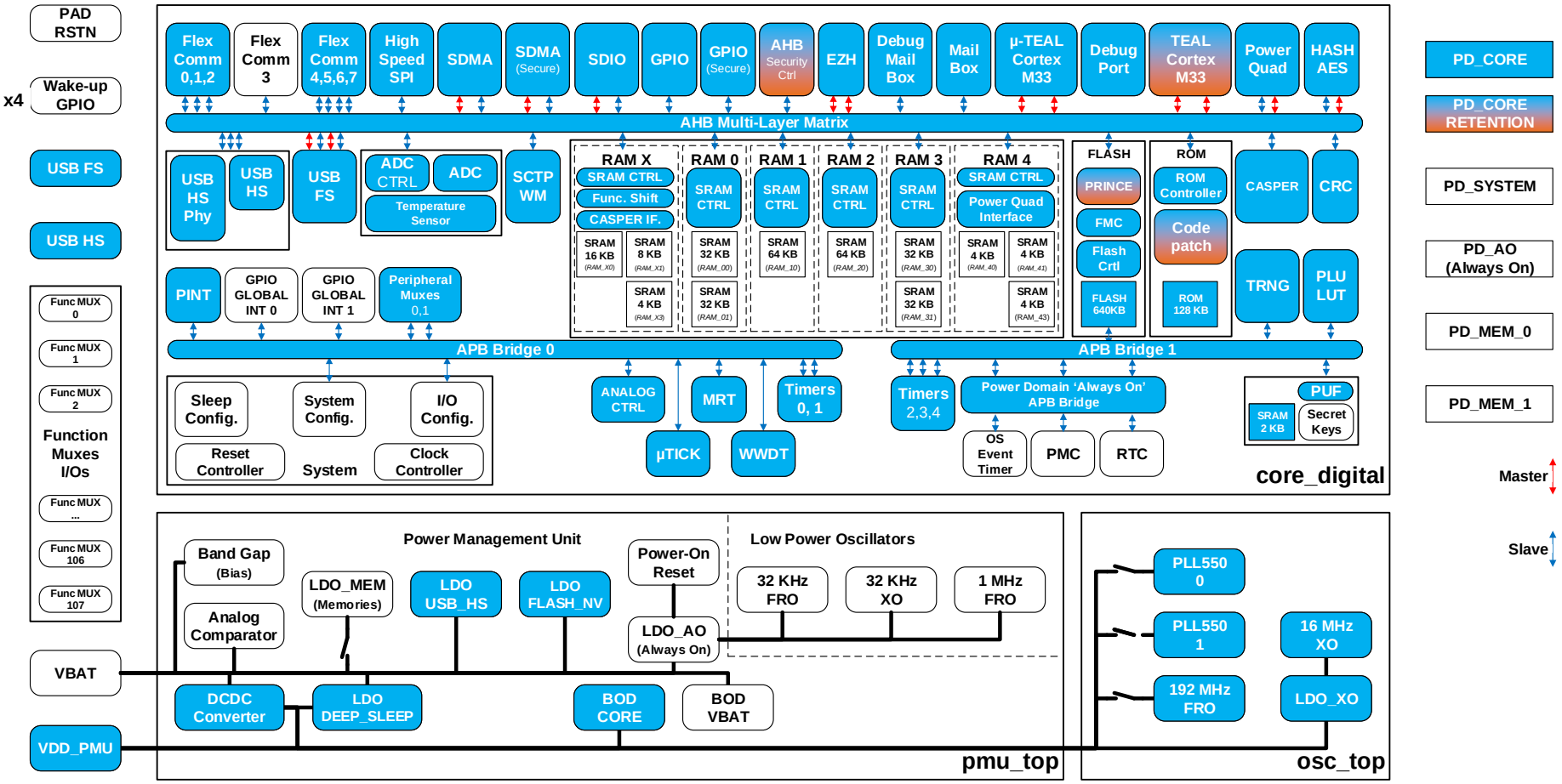
Power Domain: All together



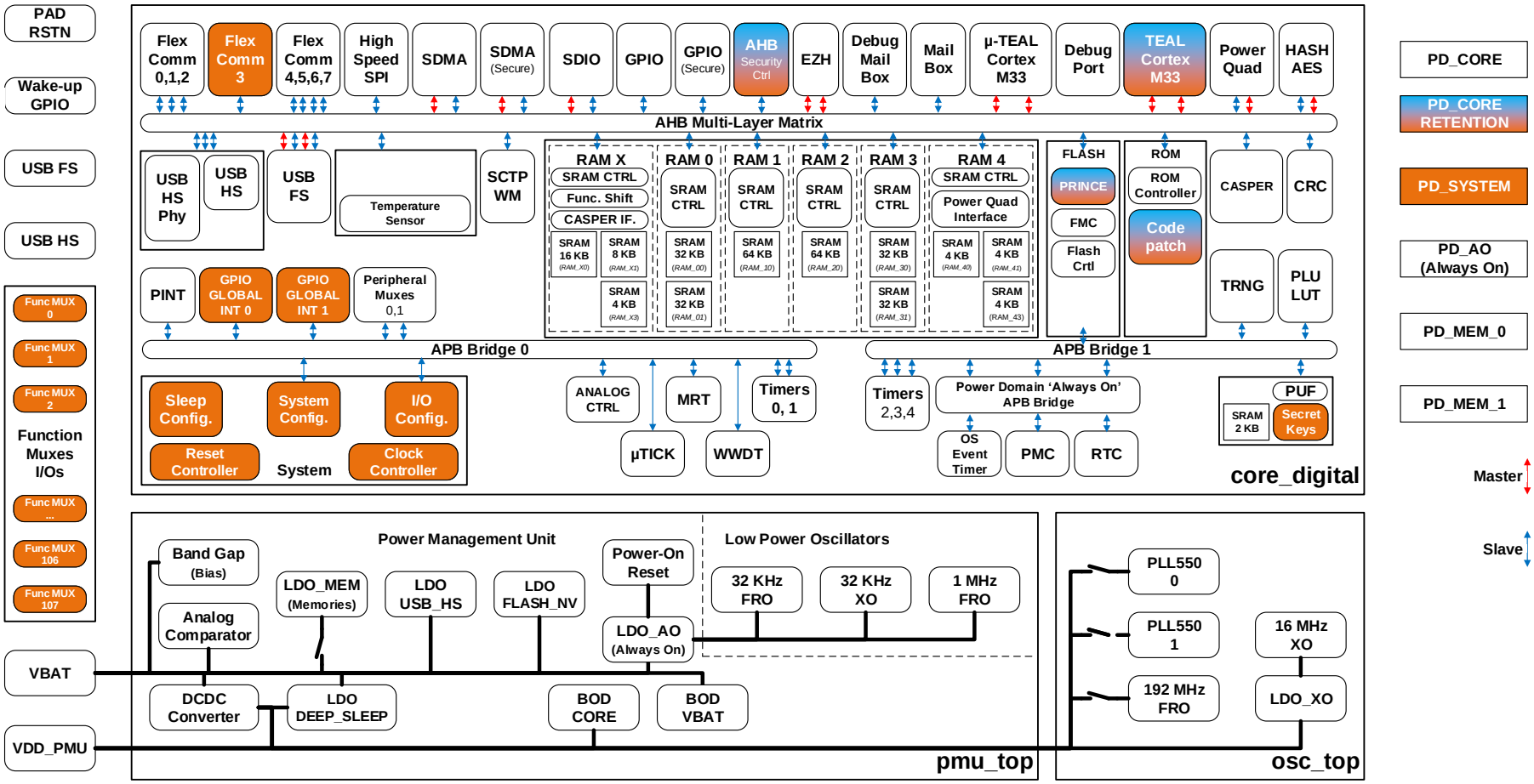
Power Domain: All together

- **The part is partitioned into 5 Power Domains:**
  - **PD_CORE**: Power Domain “Core”: most of all digital core logic (CPU0, CPU1, Multilayer Matrix, Serial Peripherals, System DMA, SDIO …)
  - **PD_SYSTEM**: Power Domain “System”: Some critical system components like Clocks Controller, Reset Controller, Syscon …
  - **PD_AO**: Power Domain “Always On”: Power Management Controller, RTC, …. This domain always has power as long as sufficient voltage is available on VBAT ([1.8V – 3.6V])
  - **PD_MEM_0**: 1st Power Domain “Memories”: 2 4-Kbytes SRAM instances
  - **PD_MEM_1**: 2nd Power Domain “Memories”: All other SRAM instances

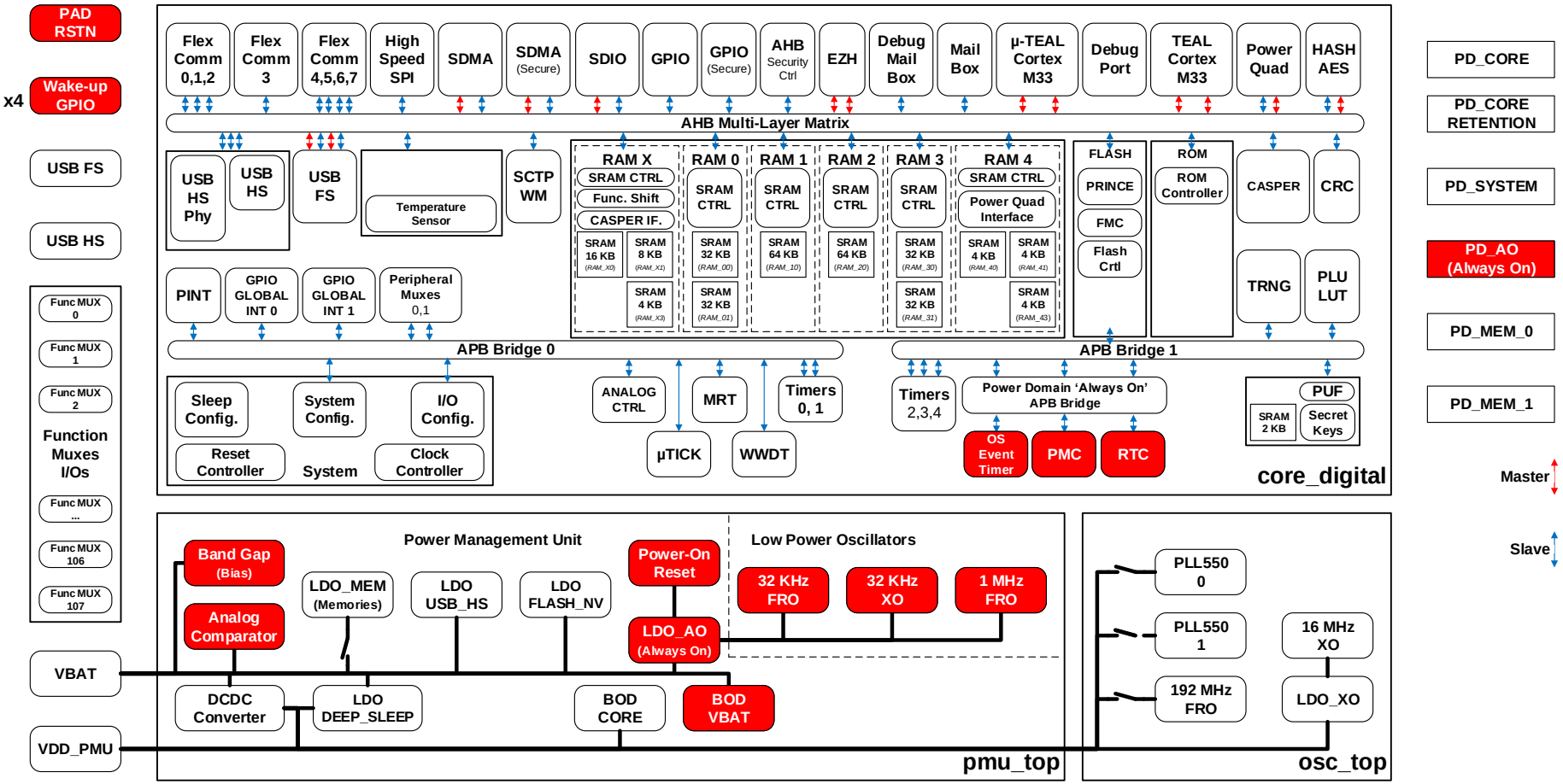
Power Domain: CORE (PD_CORE)



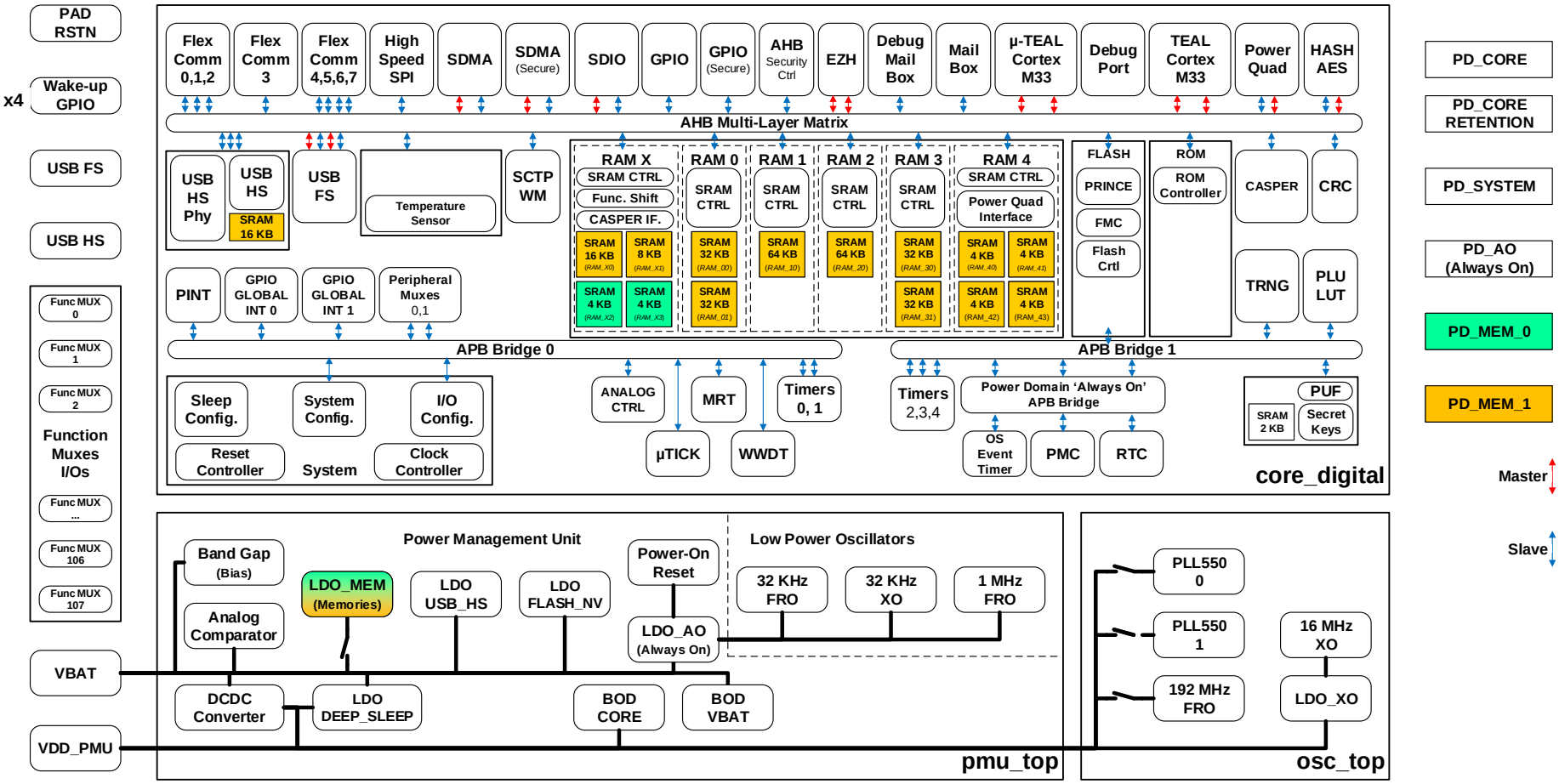
Power Domain: SYSTEM (PD_SYSTEM)



Power Domain: ALWAYS-ON (PD_AO)



Power Domain: MEMORIES (PD_MEM0 & PD_MEM1)



Power Domains supply

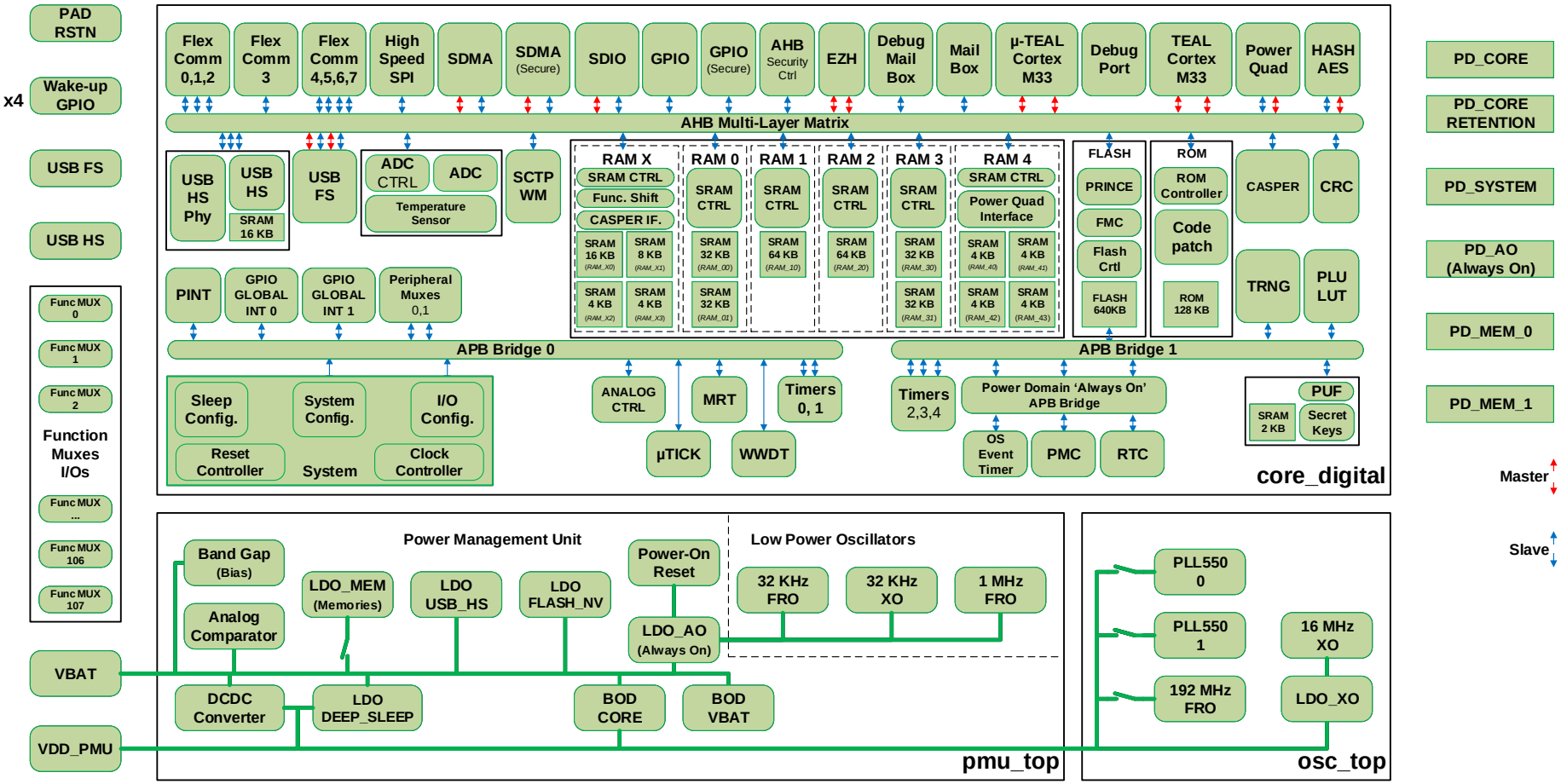
- **Power domain PD_CORE is powered:**
  - by **DCDC** in Active and Sleep modes.
  - by **LDO_DEEP_SLEEP** in Deep-sleep Mode (or by DCDC)

- **Power domain PD_SYSTEM is powered:**
  - by **DCDC** in Active and Sleep modes.
  - by **LDO_AO** in Power-down

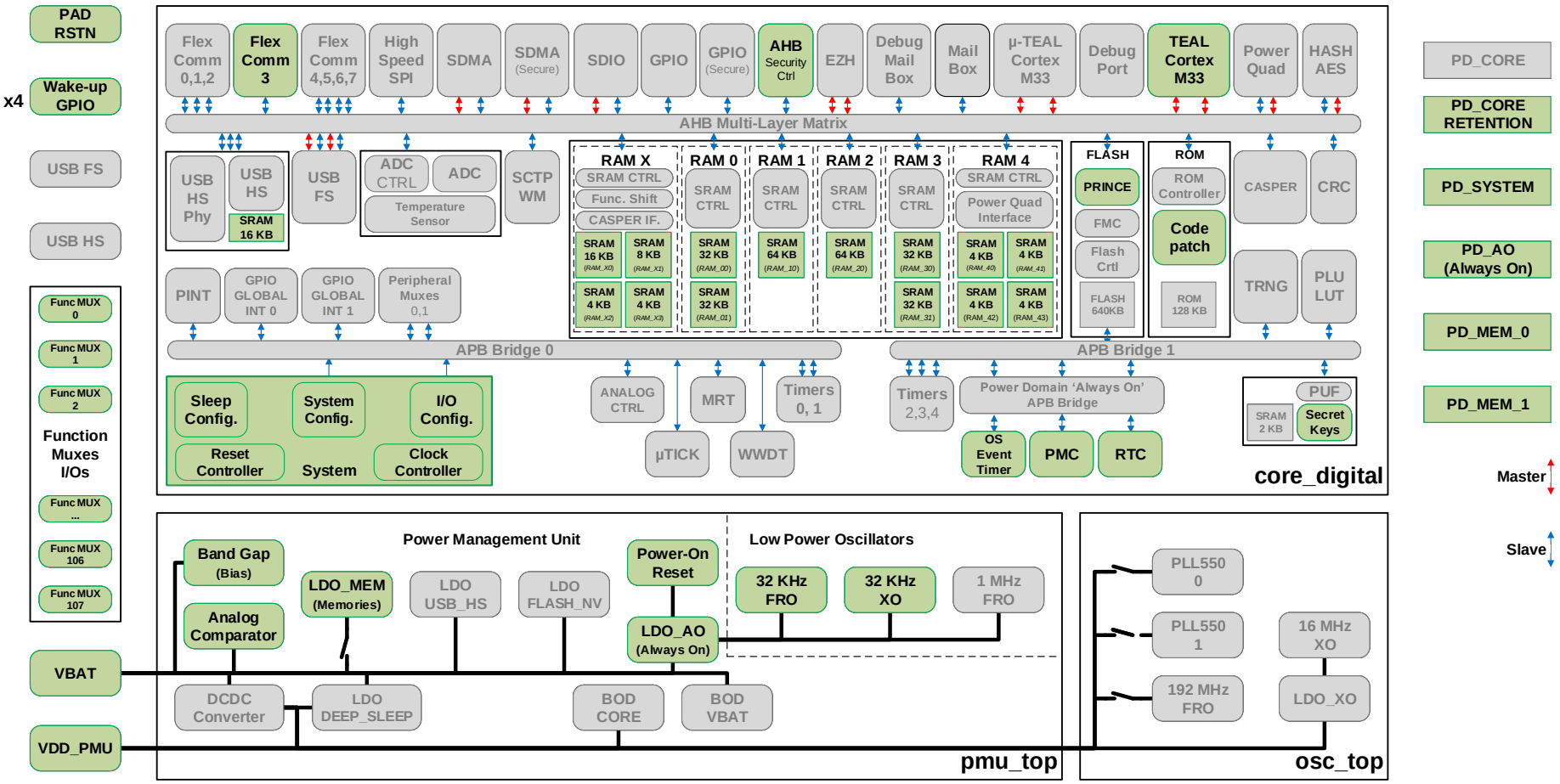
- **Power domain PD_AO is powered:**
  - by **LDO_AO** only, whatever the power mode.

- **Power domains PD_MEM_0 and PD_MEM_1 are powered:**
  - by **DCDC** in Active and Sleep power modes.
  - by **LDO_MEM** in Deep-sleep, Power-down and Deep power-down modes.

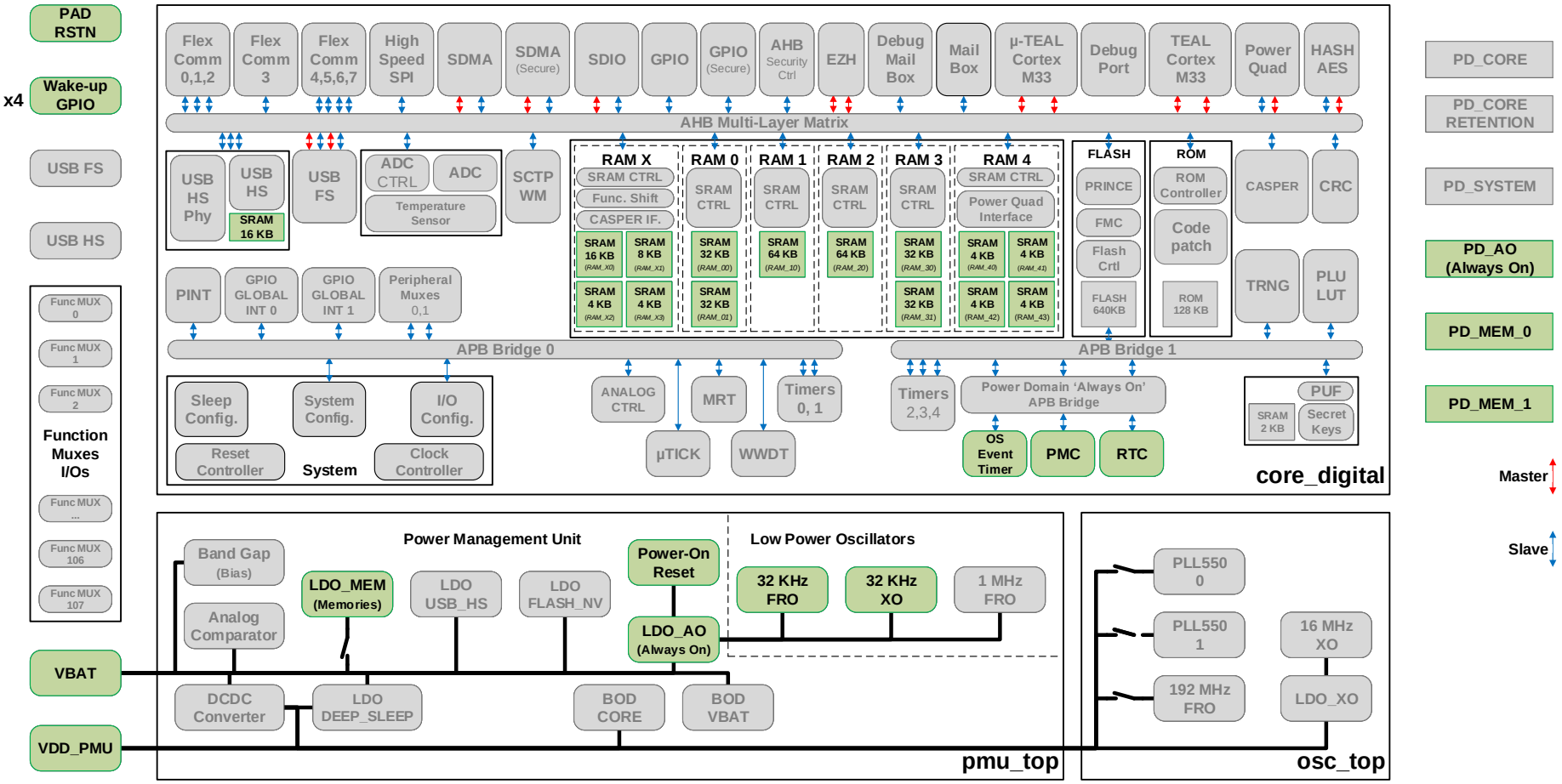
DEEP-SLEEP


PAD RSTN
x4
Wake-up GPIO
USB FS
USB HS
Func MUX 0
Func MUX 1
Func MUX 2
Function Muxes I/Os
Func MUX ...
Func MUX 106
Func MUX 107
VBAT
VDD_PMU
Flex Comm 0,1,2
Flex Comm 3
Flex Comm 4,5,6,7
High Speed SPI
SDMA
SDMA (Secure)
SDIO
GPIO
GPIO (Secure)
AHB Security Ctrl
EZH
Debug Mail Box
Mail Box
µ-TEAL Cortex M33
Debug Port
TEAL Cortex M33
Power Quad
HASH AES
AHB Multi-Layer Matrix
USB HS Phy
USB HS
SRAM 16 KB
USB FS
ADC CTRL
ADC
Temperature Sensor
SCTP WM
RAM X
RAM 0
RAM 1
RAM 2
RAM 3
RAM 4
SRAM CTRL
Func. Shift
CASPER IF.
Power Quad Interface
FLASH
PRINCE
FMC
Flash Crtl
FLASH 640KB
ROM
ROM Controller
Code patch
ROM 128 KB
CASPER
CRC
TRNG
PLU LUT
PINT
GPIO GLOBAL INT 0
GPIO GLOBAL INT 1
Peripheral Muxes 0,1
APB Bridge 0
APB Bridge 1
Sleep Config.
System Config.
I/O Config.
Reset Controller
System
Clock Controller
ANALOG CTRL
MRT
Timers 0, 1
µTICK
WWDT
Timers 2,3,4
Power Domain 'Always On' APB Bridge
OS Event Timer
PMC
RTC
PUF
SRAM 2 KB
Secret Keys
core_digital
Power Management Unit
Low Power Oscillators
Band Gap (Bias)
Analog Comparator
LDO_MEM (Memories)
LDO USB_HS
LDO FLASH_NV
Power-On Reset
LDO_AO (Always On)
32 KHz FRO
32 KHz XO
1 MHz FRO
DCDC Converter
LDO DEEP_SLEEP
BOD CORE
BOD VBAT
pmu_top
PLL550 0
PLL550 1
192 MHz FRO
16 MHz XO
LDO_XO
osc_top
PD_CORE
PD_CORE RETENTION
PD_SYSTEM
PD_AO (Always On)
PD_MEM_0
PD_MEM_1
Master
Slave

POWER-DOWN (with CPU State Retention)

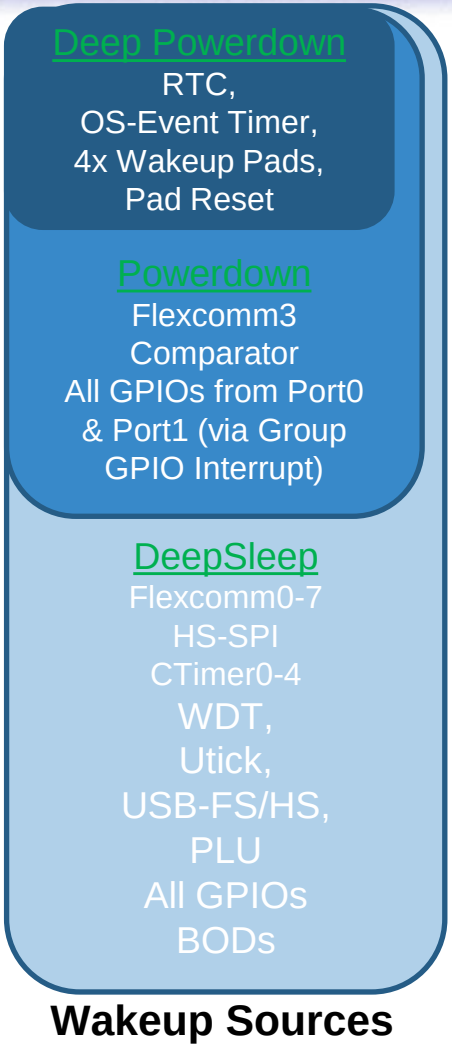


DEEP POWER-DOWN



Power View

Power Mode	Current VBAT @3V	Wake Time	Description
Active mode (Core @1.1V, @96Mhz)	32 uA/MHz	-	Core running @1.1V, 96MHz FRO
Deepsleep (All 320kB SRAM retained)	92uA	100us	Logic is powered but @0.9V, Flash/ROM off, SRAM/Padring on, Clocks off (except wakeup sources)
Powerdown w/ Retention (with 8kB SRAM retained)	3.5uA	325us	DCDC off, RAM/Padring on, CPU0 and some periph. retained, PD_System powered. PD_core is off
Deep Powerdown (with 4kB retained)	0.7uA	15ms*	Only wakeup (4x) and reset pads, and PD_AO (PMC, RTC, OS timer, PMU) are powered. Rest of the logic is off



Wakeup Sources

Peripherals

Flexcomm Serial Communication Interface

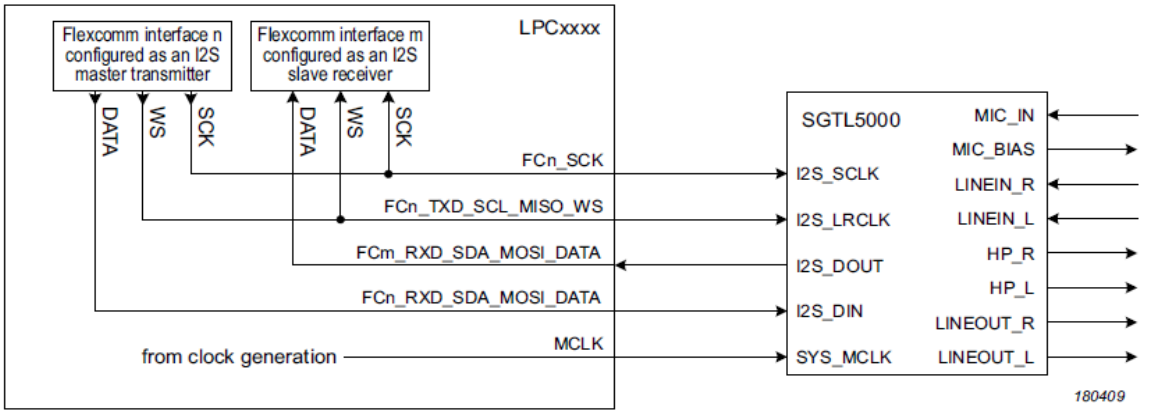
- 8x Flexcomm channels
  - SW select one among: USART, I2C, SPI, I2S
  - Multiple clock-sources, dedicated FRG per Flexcomm
  - 16Byte FIFO
  - Dedicated read and write DMA channels
- 1x High Speed SPI

*LPC553x/1x families will have four channels

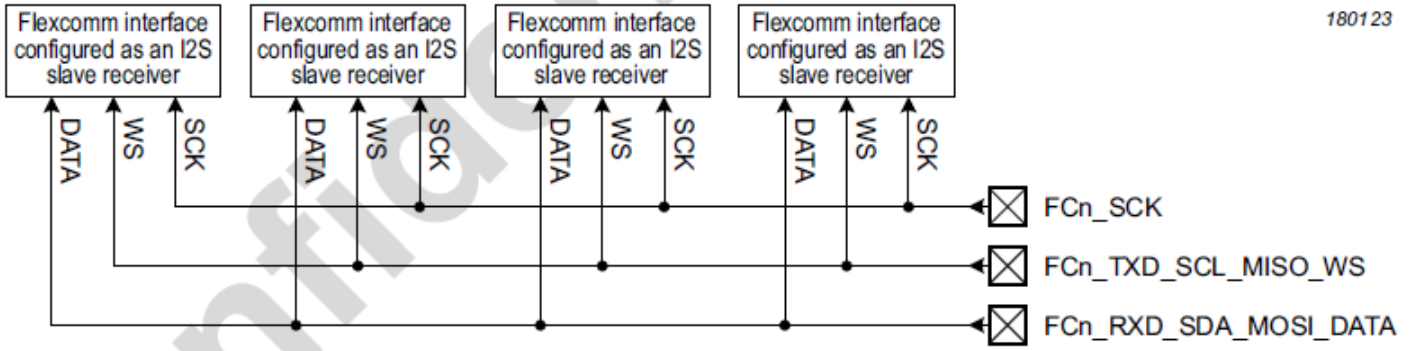
Flexcomm	Peripheral Function	Interface speed
FC0 - FC5	USART with asynchronous operation or synchronous master or slave operation.	Sync: 25MHz, Async 10MHz
	SPI master or slave, up to 4 slave selects	Master: 30MHz, Slave 20MHz
	I^2C with separate master, slave and monitor functions.	1-2MHz
	Each provide one channel pair of I^2S	24.576MHz
FC6 - FC7	Each provide one* channel pairs of I^2S function	
HS-SPI	Dedicated for High Speed SPI Up to 50MHz	Master: 50MHz, Slave 30MHz

I2S Pin-sharing

- Signal sharing allows more than one on-chip I2S interface to be connected to the same IO-pins without external board wiring
  - Allows bi-direction I2S stream with single set of I2S pins
  - Allows using multiple I2S functions in a single TDM stream with single set of I2S pins
- LPC55xx implements I2S pin-sharing among all I2S except I2S3



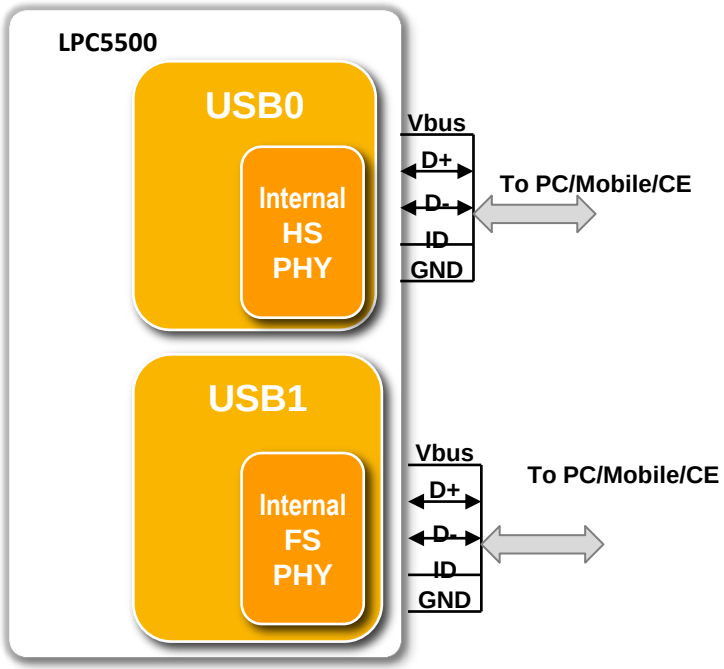
Example connection to an I2S bidirectional codec



I²S signal sharing example

USB

- USB 2.0 High-Speed Controller
  - Host/Device Modes
    - Programable selection
  - On-chip high-speed PHY
    - Can run on range of Crystal frequencies
- USB 2.0 Full-Speed Controller
  - Host/Device
    - Programable selection
  - On-chip PHY and dedicated DMA controller
  - Dedicated 3V supply pin to allow rest of the IOs to run @1.8V
  - Supports crystal-less operations in device mode

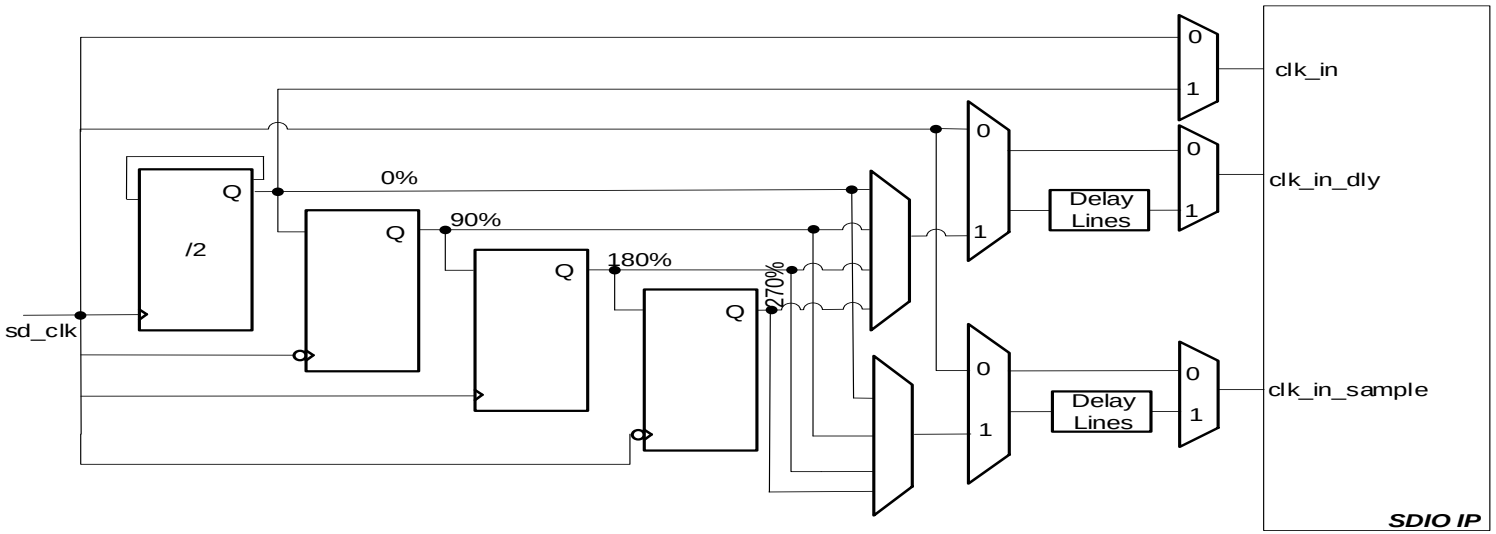


Hi-Speed and Full-Speed USB with on-board Hi-Speed PHY

With on-chip USB ROM Drivers and USB library

SDIO

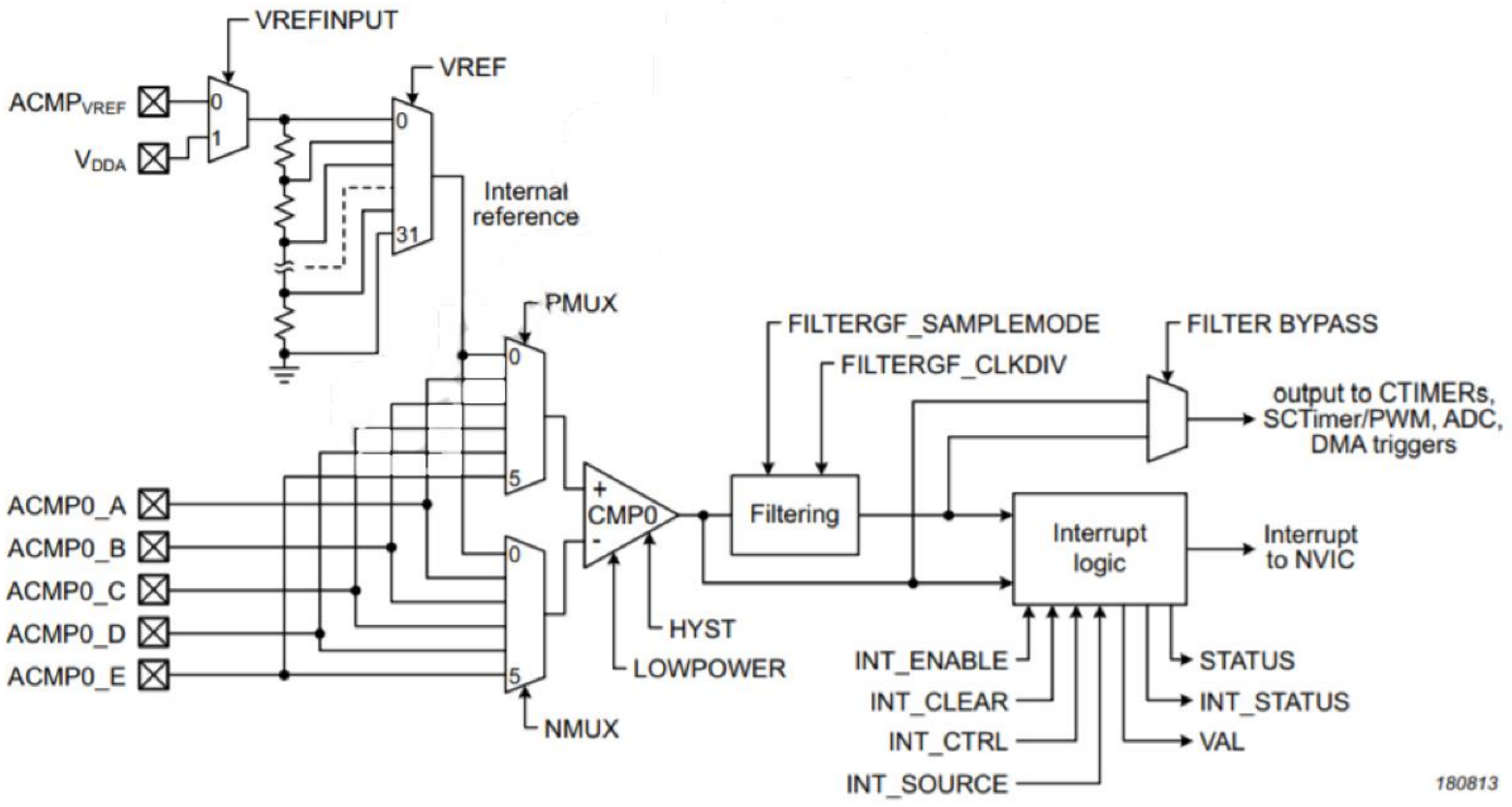
- **Secure digital input/output (SD/MMC and SDIO) card interface with integrated DMA**
  - SDIO with support for up to two cards
  - Supported cards: MMC, SDIO and CE-ATA, eMMC
  - Support for SD2.0, SDR25 (50MHz)
    - 25MBps @4bit (SDCard), 50MBps @8bit (SDIO)
  - Phase shift or Delay line for data-sampling clock delay



Analog Comparator

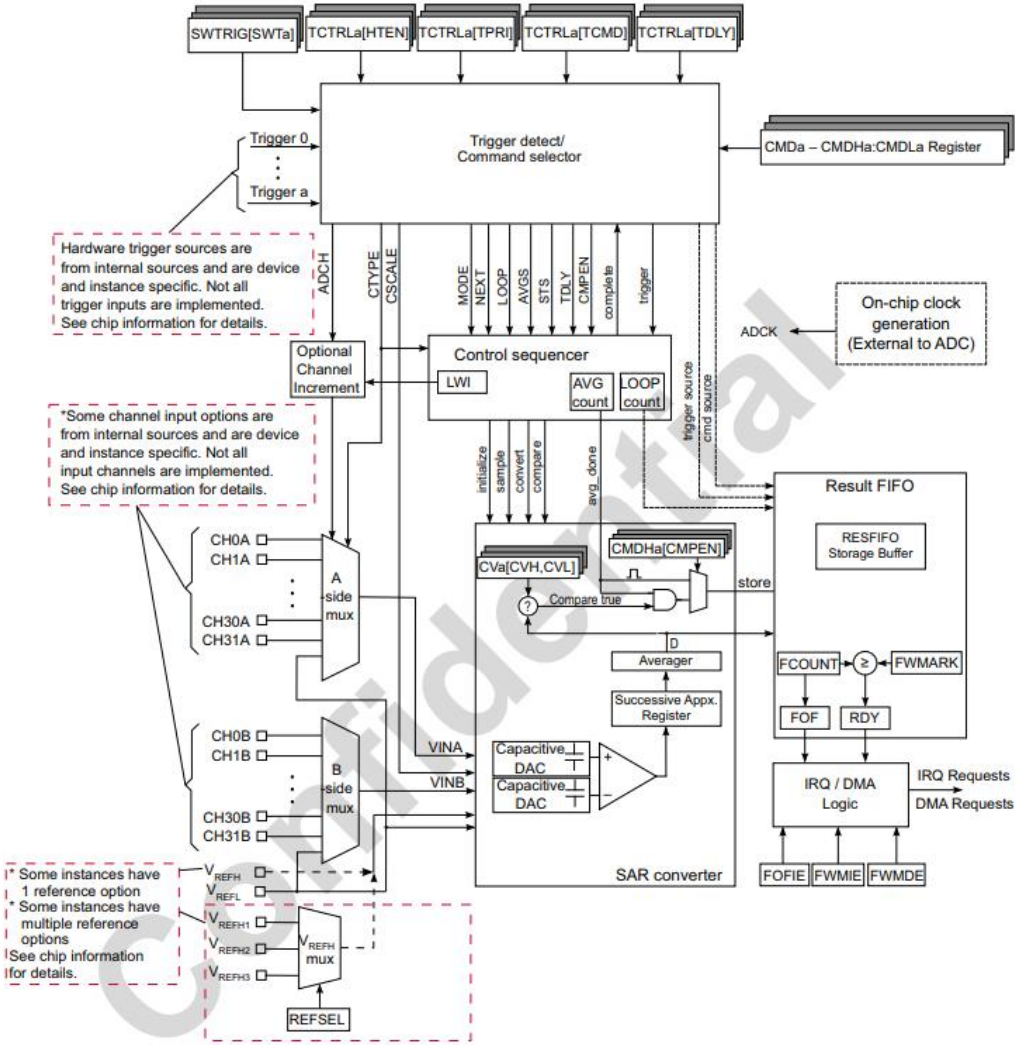
- Compares voltage levels on external pins and internal voltages
- 5 inputs are multiplexed separately to the positive voltage input and negative inputs
  - **4 inputs on 100-pin package**
- Voltage ladder source selectable between the supply pin VDD or ACMP_VREF pin
- 32-stage voltage ladder can be used as either the positive or negative input of the comparator
- Interrupt Capability
- Wakeup source

Analog Comparator



Analog to Digital Converter

- 16-bit ADC with successive approximation
  - 13-bit conversion rate @1.0 Msps, differential (12-bit for single-ended)
  - 5 differential channel pair (or 10 single ended channels 100-pin package
  - Channel pair 0-3 offer best accuracy
- Internal Temperature sensor
  - connected to internal ADC channel ADM_14
- Triggers
  - Multiple sources:
    - SCTimer/PWM
    - Ctimers Match
    - Comparator
    - GPIO Interrupt
    - ARM TXEV interrupt
  - Two configurable conversion sequences with independent triggers
- Optional automatic high/low threshold comparison and “zero crossing” detection
- Burst conversion mode for single and multiple inputs



I/O Port

- **General-Purpose Input-Output (GPIO)**
  - 64 GPIOs on 100/98-pin package, 36 GPIOs on 64-pin package
  - All GPIOs (except ISP-Select, SWD, SWCLK,TRSTN) are tri-stated at reset
  - Up to 8 GPIOs can be selected as pin interrupts
  - Registers are in system bus for fast access

- **Pin Interrupts & Pattern Engine**
  - Up to eight GPIO can be selected as external interrupt source (connected to the NVIC)
  - Pattern match engine can be used in conjunction with software to create complex state machines based on any of the eight GPIO pin inputs.
  - Pattern match engine facilities wake-up only from active and sleep modes.

- **Secure GPIO**
  - Digital I/Os that are sensitive to information leakage can be masked using SEC-GPIO_MASK0/1/2/3 registers, safeguarding incoming data on secure peripherals
  - GPIO Port0(0-31) pins have Secure GPIO as selectable pin-mux function

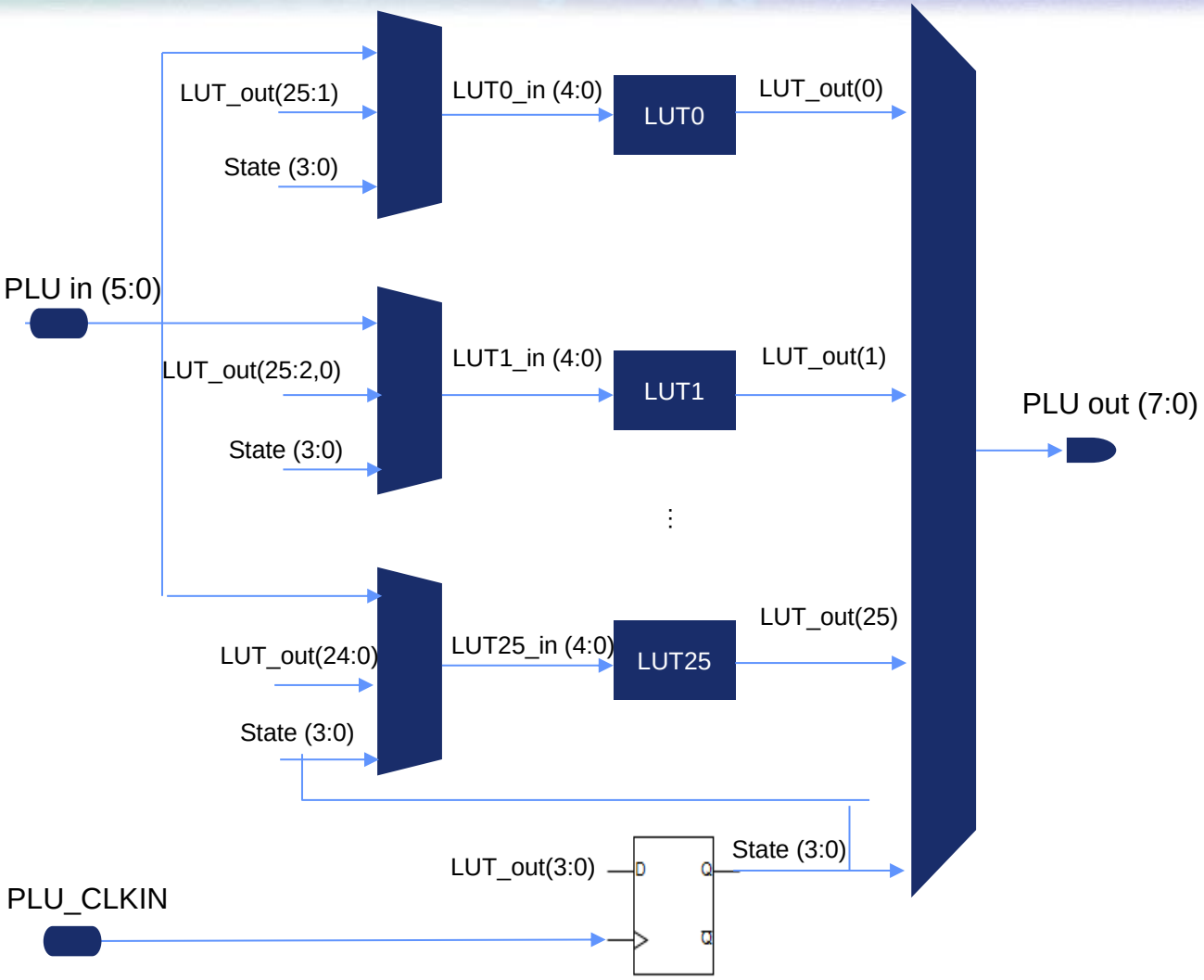
Cyclic Redundancy Check (CRC) Engine Block

- Supports three common polynomials CRC-CCITT, CRC-16, and CRC-32
  - CRC-CCITT: $X^{16} + X^{12} + X^5 + 1$
  - CRC-16: $X^{16} + X^{15} + X^2 + 1$
  - CRC-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Programmable seed number setting
- Accept any size of data width per write: 8, 16 or 32-bit
  - 8-bit write: 1-cycle operation
  - 16-bit write: 2-cycle operation (8-bit x 2-cycle)
  - 32-bit write: 4-cycle operation (8-bit x 4-cycle)

Programmable Logic Unit (PLU) – Features

- Create combinational and sequential logic networks, simple state-machine
  - 26 Look-up Table (LUT)
    - 32-bit Truth Table
  - 6 inputs + 8 outputs to pad
  - Optional clock input
    - Required for sequential network
  - Supports interrupt and low power mode wakeup (deep-sleep only)
- A GUI based PLU Tool available to facilitate the configuring work of PLU

Programmable Logic Unit (PLU) – Features



Programmable Logic Unit (PLU) – Features

- Completely independent from MCU, which only configures logic via by registers
- Up to 6 PLU inputs from I/O pads
- Up to 8 outputs to I/O pads
- Up to 26 Look up tables (LUTs)
- Up to 4 state flip flops
- Clock input from external I/O pad

- Feedback to MCU via I/O pads or through switch matrix

Programmable Logic Unit (PLU) – Uses

- Some potential functionality to implement in the PLU:
  - Integration of discrete logic device(s) to save cost and area
  - Event detection
  - Simple state machines
  - Stepper motor control
  - Generation of specialized waveforms / protocols in conjunction with other peripherals
- Benefits (vs function implementation on conventional MCU)
  - Low latency
  - Low power (can operate while MCU is in deep sleep)
  - Save MCU processing power for other tasks

Programmable Logic Unit (PLU) – Uses

- The PLU is used to create small combinatorial sequential logic operating networks.
  - The logic operation can be programmed by software.
- Object resource:
  - 26 LTUs (Look-up Table, or so-called Logic True Value Table), per LUT has a 5-bit input port (each bit is connected to a input line) and the values can be indexed in the range of 0~31.
  - 4 Flip-Flops driven by an external PLU_CLKIN clock.
  - 6 PLU inputs from external pins.
  - 8 PLU output to external pins.
- Support low power mode including deep-sleep and power-down mode.
- (Optional) A GUI tool, PLU Tool, is available to facilitate the configuring work of PLU.

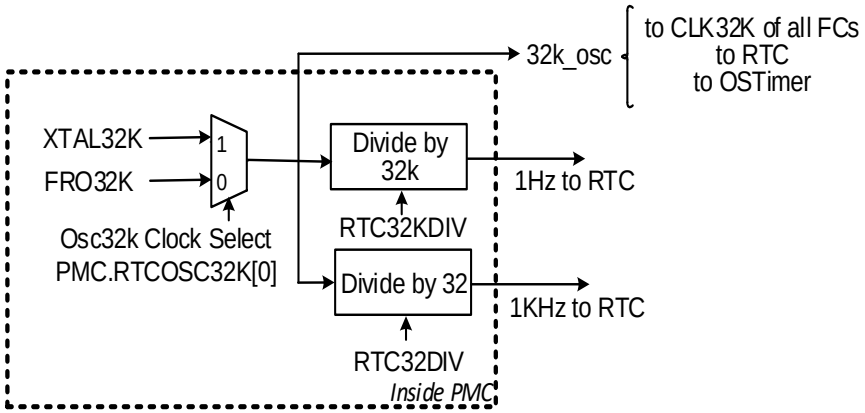
Programmable Logic Unit (PLU) – Advantages

- Simple glue logic to connect MCU with external components and replace these 7400 series.
- State machine design using Flip-flop
- Address decoder
- Pattern match
- Low-power application. Many peripherals are shut off in deep-sleep and power-down mode, PLU is not.
- PLU is Programmable!!! It can be reprogrammed and reused.
- Seamless connection using SWM and PLU. e.g. connect MCU standard CTIMER MATx output to PLU input, PLU does its magic, connect PLU output to the CTIMER CAPx input.

Always-ON Timers

- Real-time clock (RTC)
  - 32-bit, 1s resolution
  - Runs on 1Hz clock source
  - Free running Timer as continuous time-base for the system
  - 1ms resolution for wakeup

- OS-Event Timer
  - 42-bit, sub-second timer
  - 2x Match and 2x Capture registers
  - Runs on 32kHz clock source



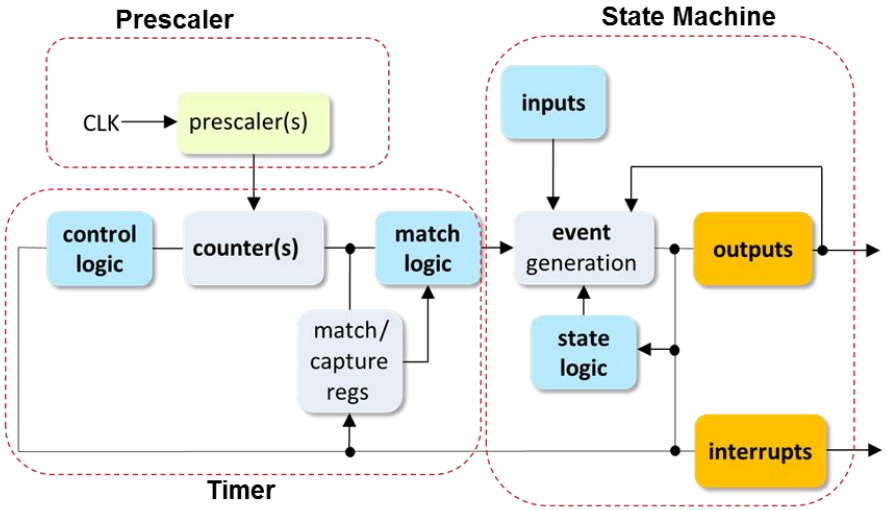
- In the always-on power domain
  - Available in any reduced power modes
  - Wake-up source from all low power modes including deep power-down
- Clocked by the 32 kHz on-chip oscillator or 32kHz External crystal

State Configurable Timer

- More programmability than Ctimer.
- Single 32-bit counter or Dual 16-bit counters
- It‘s a timer, enhanced by
  - Input capture events
  - Signal output capabilities
  - Interrupt and DMA events
  - A state machine which defines the behavior of counter, outputs, interrupts, DMA in a flexible way, over more than one cycle of the counter
  - A lot of interconnections between all these
- **LPC55xx configuration**
  - 8 inputs
  - 10 outputs
  - 16 match/capture registers
  - 16 events
  - 32 states

It’s a timer
- Input Capture
- Output Compare
- PWMs

…**with a state machine**



General Purpose Timers

- 5x general purpose continuous timer/counter (Ctimer, 32-bit)
  - **Counter engine can run on a clock independent of system clock**
  - Capture, Match and Auto Match Reload function
  - Wake-up source from the Deep-sleep

- Windowed watchdog timer (WWDT, 24-bit)
  - Running from the FRO1M
  - Can generate system-reset
  - Wake-up source from the Deep-sleep

- Micro-Tick Timer (UTICK, 31-bit)
  - Running from the FRO1M
  - Wake-up source from the Deep-sleep

- Multi-Rate timer (MRT, 24-bit)
  - Timer with four independent channels
  - Repeat interrupt, one-shot interrupt, and one-shot bus stall modes

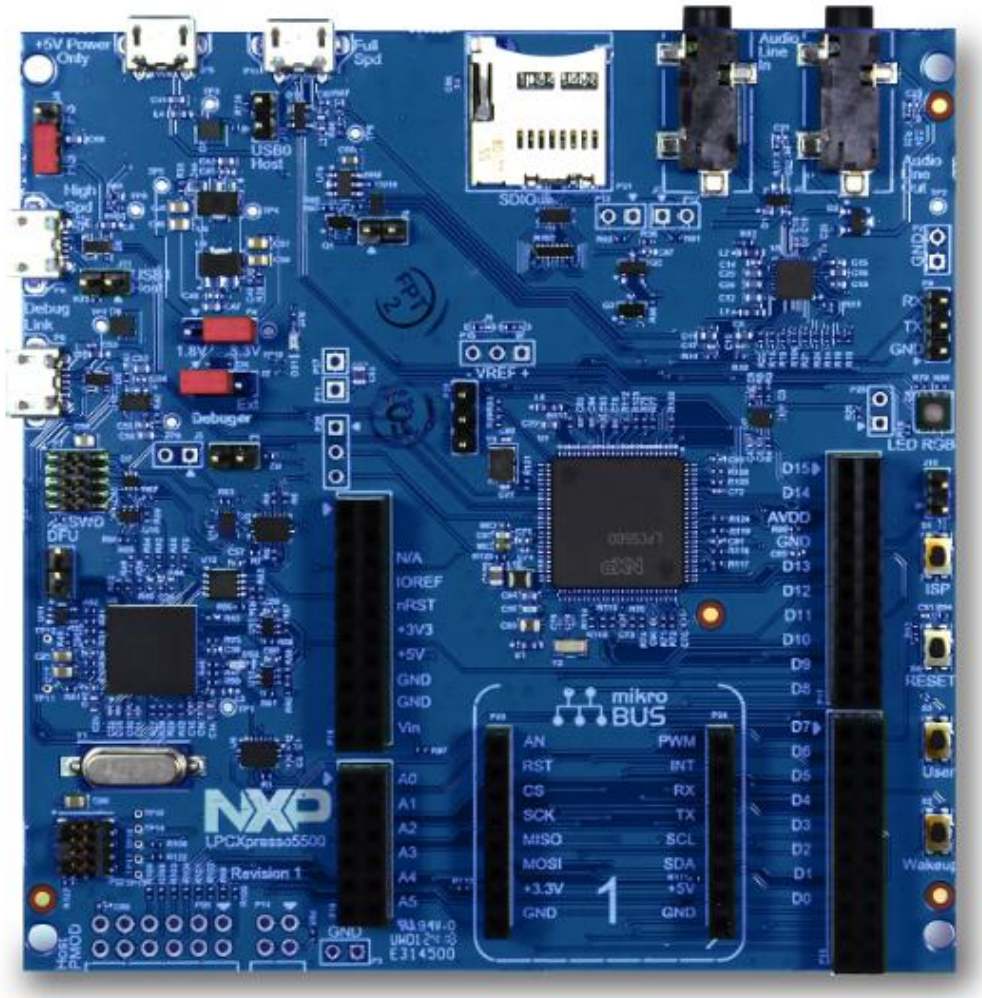
Enablement (Development)

LPCXpresso55S69 Development board

- **LPC55S69 Dual Cortex-M33 core processor @ 100MHz**
- **Onboard, high-speed USB, Link2 debug probe with CMSIS-DAP and SEGGER J-Link protocol options**
  - UART/SPI bridging from LPC55S69 to USB via the onboard debug probe
  - Support for external debug probe
- **Various on-board interfaces and components**
  - RGB user LED
  - Reset, ISP, User/Wakeup and user buttons
  - NXP MMA8652FCR1 accelerometer
  - Micro SD card slot
  - Stereo audio codec with a line in/out
  - High / full speed USB port with micro A/B connector for the host or device functionality
- **Flexible expansion - Arduino, Mikroe & Pmod – provide developers easy access to a broad range of connectivity and sensors options**

LPCXpresso55S69 Development board

LPC55S69-EVK



SDK IDE CFG

ARM KEIL Microcontroller Tools

IAR SYSTEMS

Enablement

Runtime Software

NXP Solutions:

MCUXpresso SDK

- Drivers
- System Services
- FreeRTOS
- USB
- Filesystem

RTOS, Middleware Partners:

amazon FreeRTOS

Zephyr™

MBED OS

Phoenix-RTOS

Software Tools

IDE / Toolchains / Configurations:

MCUXpresso IDE

MCUXpresso Configuration Tools

ARM KEIL Microcontroller Tools

IAR SYSTEMS

SEGGER www.segger.com

ARM mbed

atollic

Rowley Associates

Hardware Tools

Evaluation Kits:

LPCXpresso55S69 Development board

Debug Probes

SEGGER www.segger.com

LAUTERBACH DEVELOPMENT TOOLS

iSYSTEM

Comprehensive frameworks and solutions for low-power, connected, and secure embedded systems

Industry leading IDE support and intuitive software configuration tools to accelerate application development

Low cost hardware platforms for evaluation and application development. Partner solutions for hardware debugging solutions

LPC5500 Real Time Operating Systems

- **Amazon FreeRTOs**
  - FreeRTOS is a royalty-free mini Real-Time Kernel with a very small footprint. Free download – www.freertos.org
  - Ported to 30+ architectures and 17 toolchains
  - Example projects for many architectures and IDEs

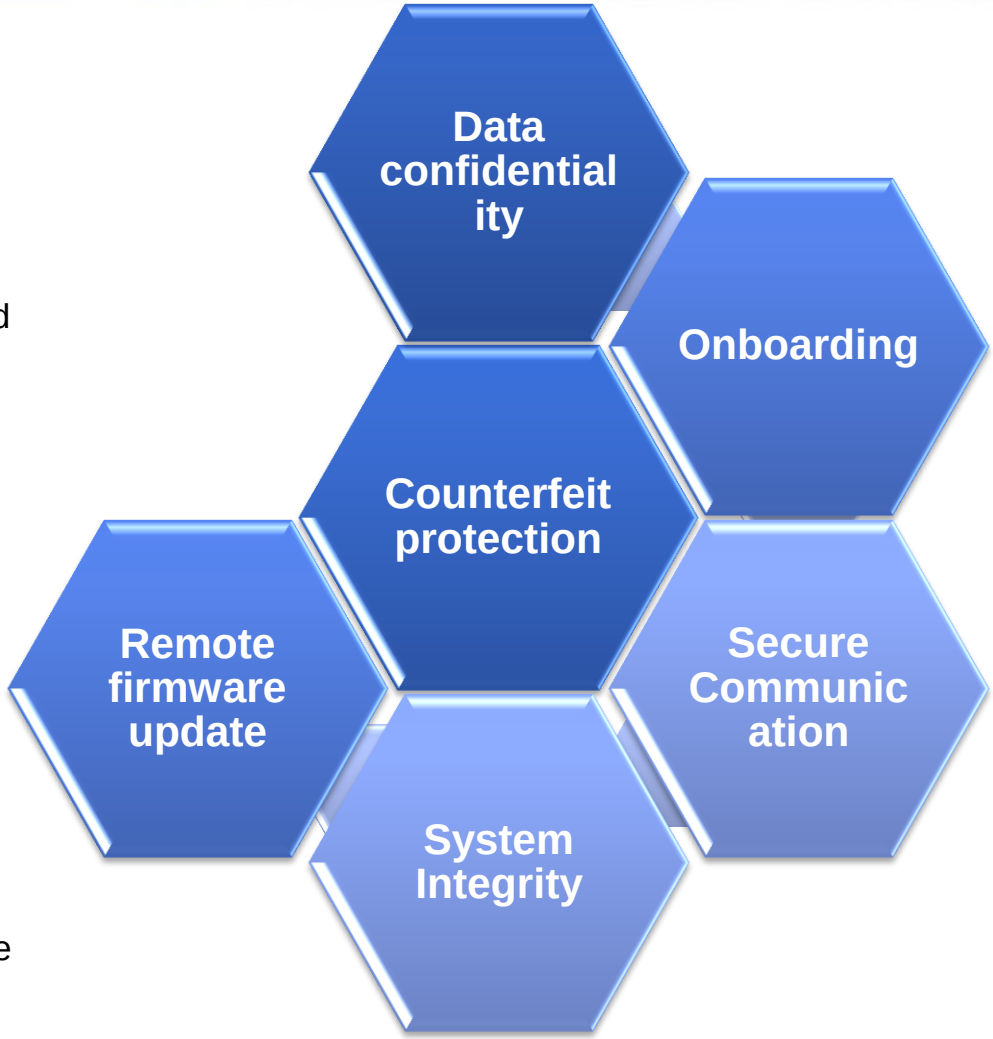
- **Zephyr**
  - Open source real time operating system – www.zephyrproject.org
  - Complete, fully integrated, highly configurable, modular for flexibility
  - Built with safety and security in mind

- **Mbed OS**
  - Free, open-source embedded operating system designed specifically for the “things” in the IoT
  - OS Features: Modular, secure and connected
  - Online IDE, toolchains and compatible with other IDEs

- **Phoenix OS**
  - Open-source, microkernel-based, real time operating system for resource constrained devices
  - OS for software defined solutions

Essential Security Goals enabled with LPC5500

- **Counterfeit protection**
  - Every device has a unique identity that can not be reproduced by an attacker
- **Onboarding**
  - Shared credentials between the end device and the back end system
- **System integrity**
  - Trust in the functionality provided by the end device
- **Secure communication**
  - Cryptography applied to the communications for the device
- **Data confidentiality**
  - Protection of data in the device
- **Remote firmware update**
  - Safe updating of the software on the end device



Referințe

https://www.arm.com/products/silicon-ip-cpu/cortex-m/cortex-m33

https://www.nxp.com/