Scopo della lezione

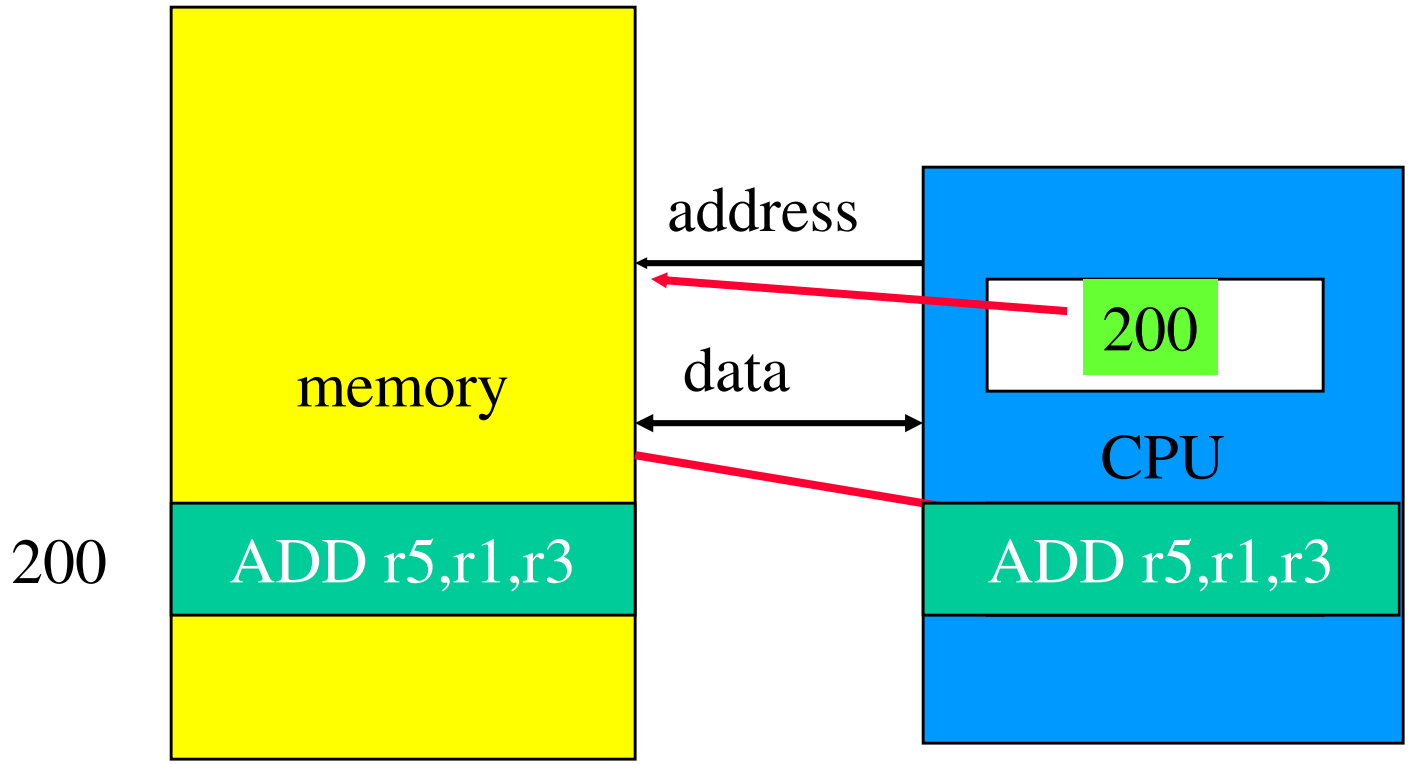
- Analizzare i tipi di “macchine”
- Indirizzamento e memorie
- Tipi di dato
- Little endian e big endian
- Indirizzamento logico e fisico
- Comprendere la struttura del micro

Von Neumann architecture

- La memoria contiene dati e istruzioni.
- La CPU “fetcha” istruzioni dalla memoria e legge e scrive dati sempre sulla stessa.

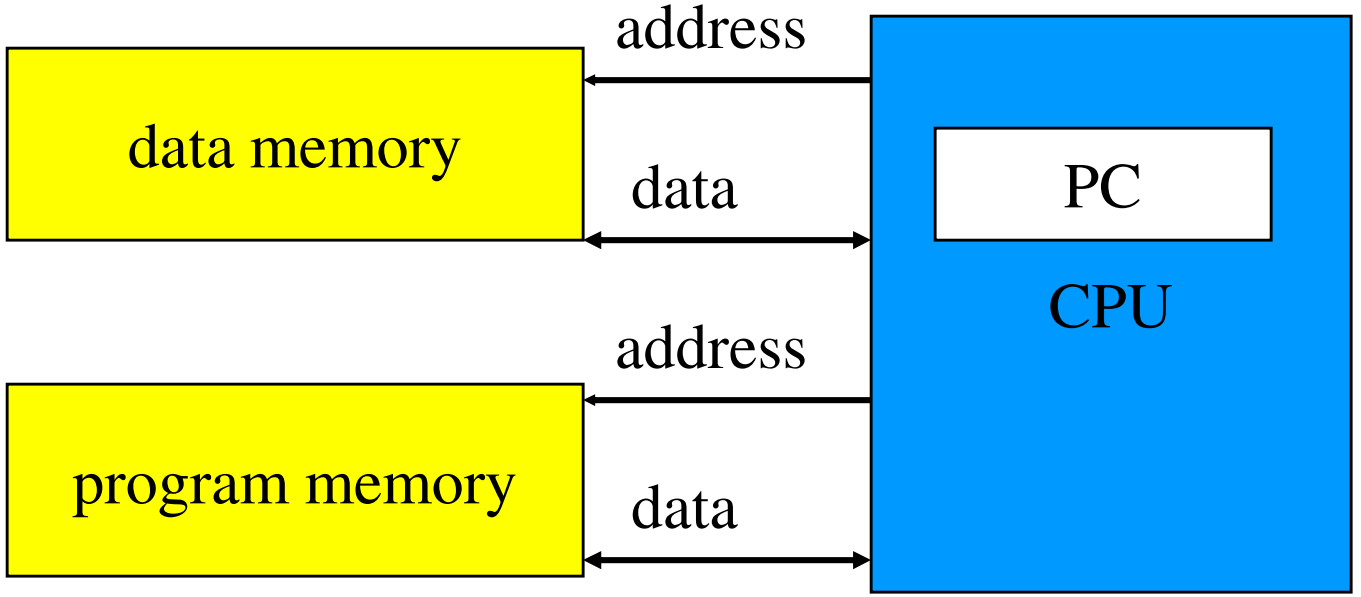
- I registri della CPU lavorano per l’esecuzione del programma: program counter (PC), instruction register (IR), registri general-purpose, ecc.

CPU + memoria comune



Harvard architecture

Caratterizzata da memoria dati e di programma separate



von Neumann vs. Harvard

- Harvard non può utilizzare codice che si auto-modifica, quindi è più sicura.
- Harvard permette 2 fetch alla memoria simultanei
- Molti DSP sono progettati secondo lo schema Harvard architecture a causa di necessità di data-streaming:
  - maggiore memory bandwidth;
  - Bandwidth maggiormente predicibile, a causa dello schema più rigido.

RISC vs. CISC

- Complex instruction set computer (CISC):
  - Permette molti modi di indirizzamento;
  - Permette un più elevato numero di operazioni codificate.
- Reduced instruction set computer (RISC):
  - Permette solamente le load/store di dati e le fetch di codice;
  - Permette di utilizzare istruzioni in pipeline.

Caratteristiche dell'Instruction set

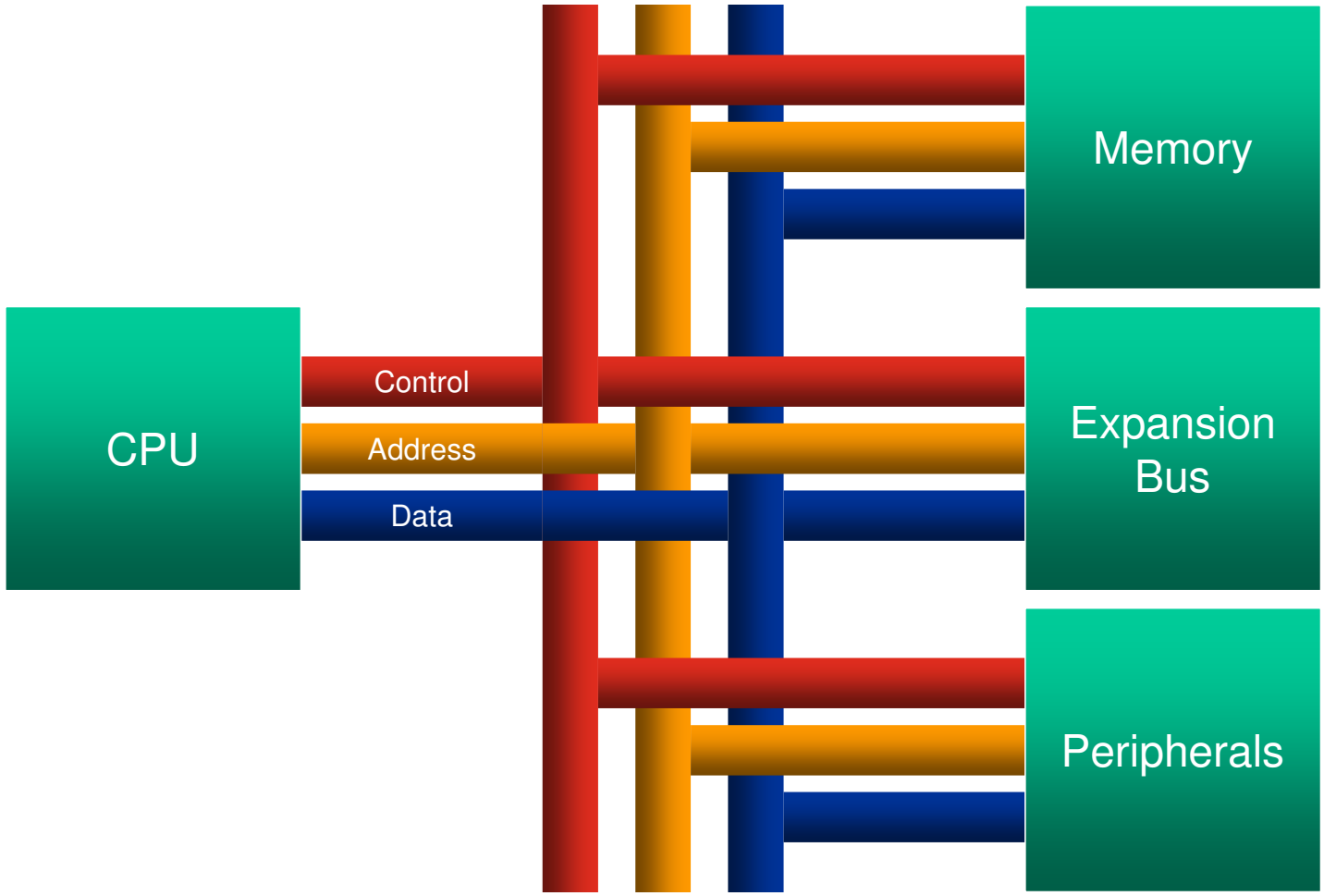
- Lunghezza fissa (RISC) vs. lunghezza variabile (CISC).
- Modi di indirizzamento.
- Numero di operandi permessi per istruzione (ad esempio il motorola 698332 permette le tblu e tblun che sono operazioni di index/ratio su vettori che rendono index e ratio in una sola istruzione; come parametri hanno il puntatore al vettore e la grandezza su cui eseguire idx/ratio).
- Tipi di operandi.

Implementazioni diverse dello stesso core

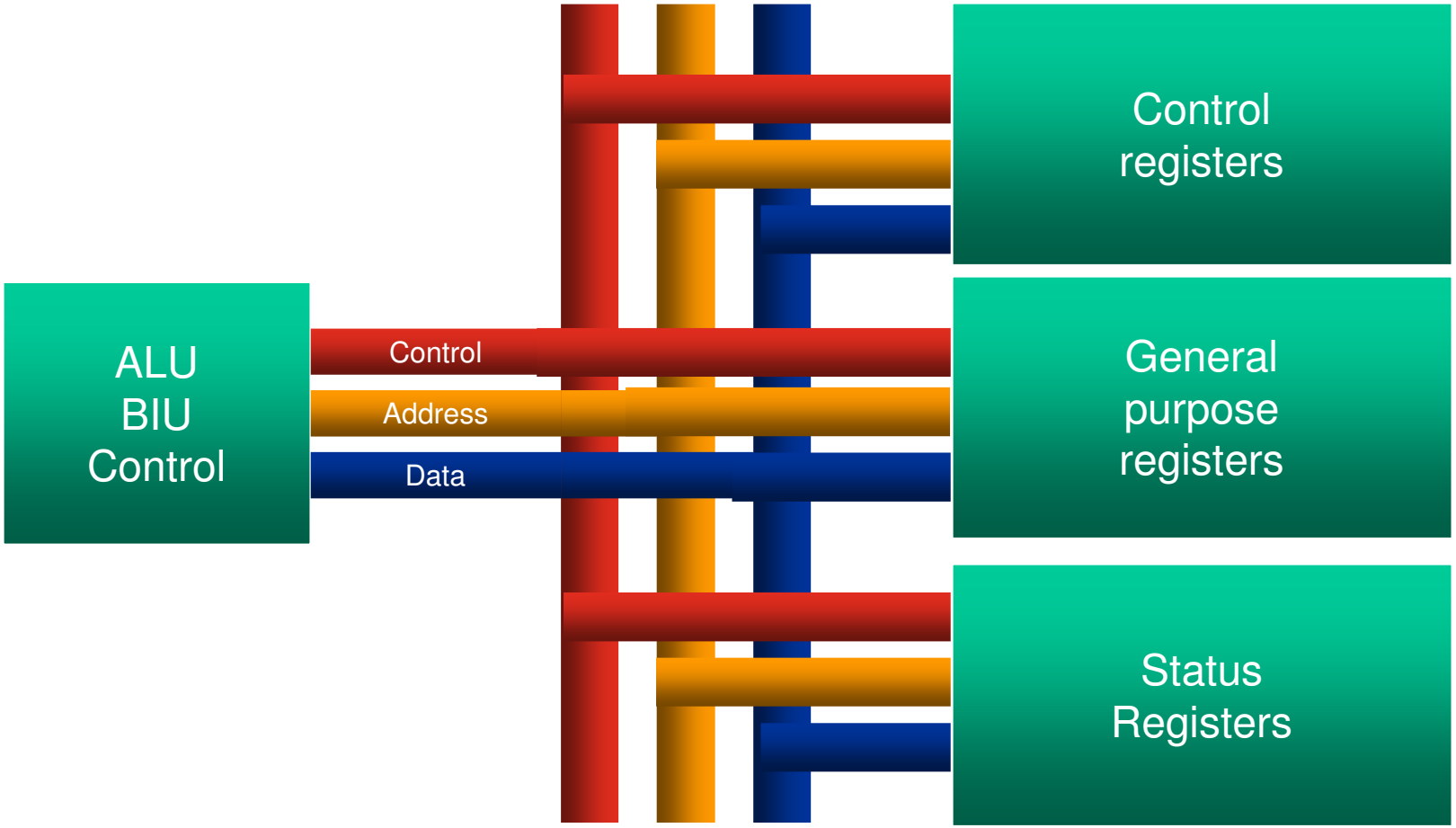
- Architetture ben riuscite presentano implementazioni diverse:
  - Con diverse frequenza di clock;
  - Diverse capacità (parallelismo) di bus;
  - Diversi tagli di memoria e di cache (ove presente);
  - etc.

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8-bit/16-bit	Ext Bus	Max Fosc (MHz)
	Bytes	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Master I^2C				
PIC18F6520	32K	16384	2048	1024	52	12	5	Y	Y	2	2/3	N	40
PIC18F6620	64K	32768	3840	1024	52	12	5	Y	Y	2	2/3	N	25
PIC18F6720	128K	65536	3840	1024	52	12	5	Y	Y	2	2/3	N	25
PIC18F8520	32K	16384	2048	1024	68	16	5	Y	Y	2	2/3	Y	40
PIC18F8620	64K	32768	3840	1024	68	16	5	Y	Y	2	2/3	Y	25
PIC18F8720	128K	65536	3840	1024	68	16	5	Y	Y	2	2/3	Y	25

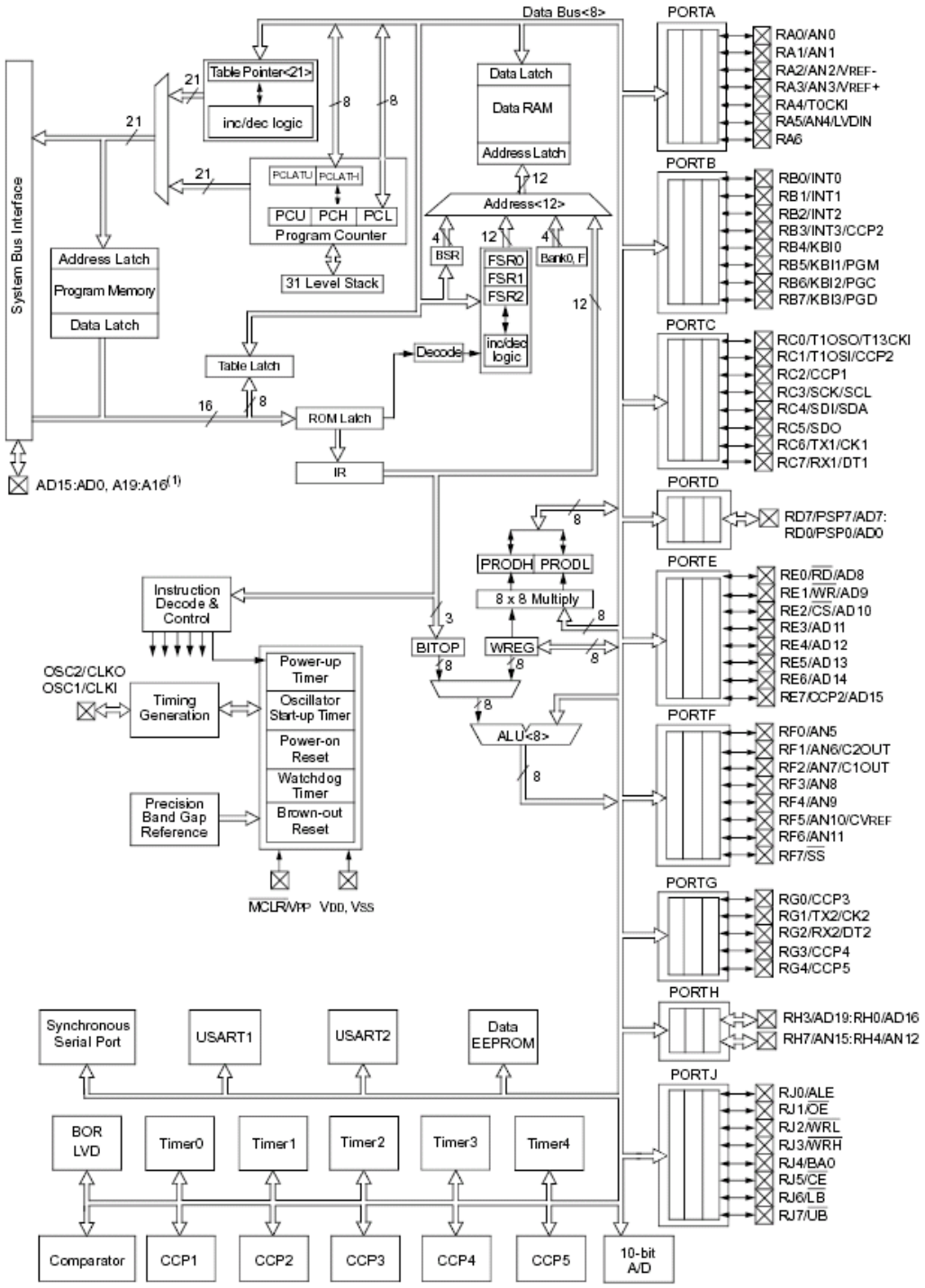
Struttura di un PC



Struttura di un microprocessore



PIC18F8x20


Data Bus<8>
PORTA
RA0/AN0
RA1/AN1
RA2/AN2/VREF-
RA3/AN3/VREF+
RA4/T0CKI
RA5/AN4/LVDIN
RA6
PORTB
RB0/INT0
RB1/INT1
RB2/INT2
RB3/INT3/CCP2
RB4/KBI0
RB5/KBI1/PGM
RB6/KBI2/PGC
RB7/KBI3/PGD
PORTC
RC0/T1OSO/T13CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX1/CK1
RC7/RX1/DT1
PORTD
RD7/PSP7/AD7:
RD0/PSP0/AD0
PORTE
RE0/RD/AD8
RE1/WR/AD9
RE2/CS/AD10
RE3/AD11
RE4/AD12
RE5/AD13
RE6/AD14
RE7/CCP2/AD15
PORTF
RF0/AN5
RF1/AN6/C2OUT
RF2/AN7/C1OUT
RF3/AN8
RF4/AN9
RF5/AN10/CVREF
RF6/AN11
RF7/SS
PORTG
RG0/CCP3
RG1/TX2/CK2
RG2/RX2/DT2
RG3/CCP4
RG4/CCP5
PORTH
RH3/AD19:RH0/AD16
RH7/AN15:RH4/AN12
PORTJ
RJ0/ALE
RJ1/OE
RJ2/WRL
RJ3/WRH
RJ4/BA0
RJ5/CE
RJ6/LB
RJ7/UB
System Bus Interface
AD15:AD0, A19:A16(1)
Table Pointer<21>
inc/dec logic
PCLATU
PCLATH
PCU
PCH
PCL
Program Counter
31 Level Stack
Address Latch
Program Memory
Data Latch
Table Latch
ROM Latch
IR
Decode
Data Latch
Data RAM
Address Latch
Address<12>
BSR
FSR0
FSR1
FSR2
Bank0, F
inc/dec logic
PRODH
PRODL
8 x 8 Multiply
BITOP
WREG
ALU<8>
Instruction Decode & Control
Timing Generation
Power-up Timer
Oscillator Start-up Timer
Power-on Reset
Watchdog Timer
Brown-out Reset
Precision Band Gap Reference
OSC2/CLKO
OSC1/CLKI
MCLR/VPP
VDD, VSS
Synchronous Serial Port
USART1
USART2
Data EEPROM
BOR LVD
Timer0
Timer1
Timer2
Timer3
Timer4
Comparator
CCP1
CCP2
CCP3
CCP4
CCP5
10-bit A/D

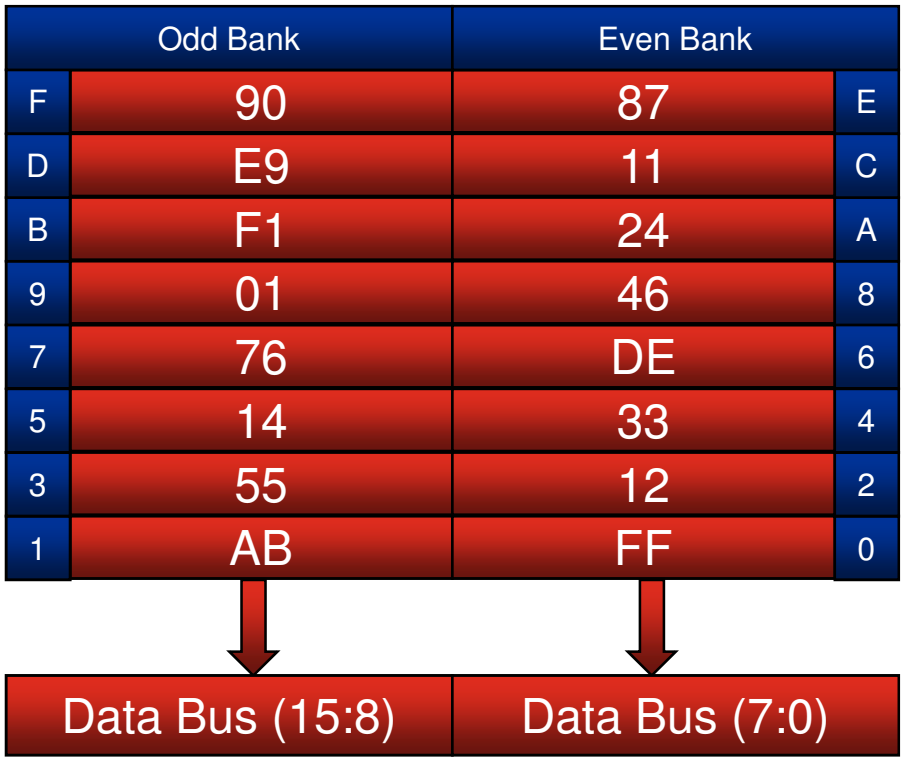
Basics

- La memoria dei PIC18F8x20 è byte-addressable
  - Ogni volta che un indirizzo è emesso sull'address bus, viene specificata una precisa locazione di memoria
- Il Byte è la "basic memory unit"
- Viene anche data la possibilità di leggere/scrivere più di un byte con una sola operazione, ovvero con un solo *ciclo di bus*
  - 16-bit: word, 2 byte consecutivi
  - 32-bit: doubleword, 4 byte consecutivi (questa operazione però non è possibile con i PIC, è invece possibile con gli ARM, I 68000 Motorola, I Pentium ecc…).

Memoria Logica vs. Fisica

- La memoria logica è la "rappresentazione" della memoria vista dal programmatore
  - Un grande byte-addressable array di byte
  - Si possono leggere/scrivere byte, word o doubleword
  - Non ci si preoccupa di come I dati o il programma è letto (fetched=andare a prendere) dalla memoria, si vede solo il risultato dell'accesso alla memoria a 1, 2, or 4 byte
- Memoria Fisica
  - E' l'organizzazione fisica delle celle di memoria, che non è visibile al programmatore
  - L'Unità di accesso alla memoria fisica è uguale alla larghezza del data bus.
  - Ad esempio 16 bit in PIC18F8x20, 32 bit in 80386 e nel motorola 68000, 64 bit nel Pentium

PIC18F8x20 memoria fisica



- Una “read” al byte di address 0: rende=FF
- Una “read” a word dall’address 0: restituisce=ABFF
- Una “read” di una doubleword dall’ address 0: restituisce=5512ABFF

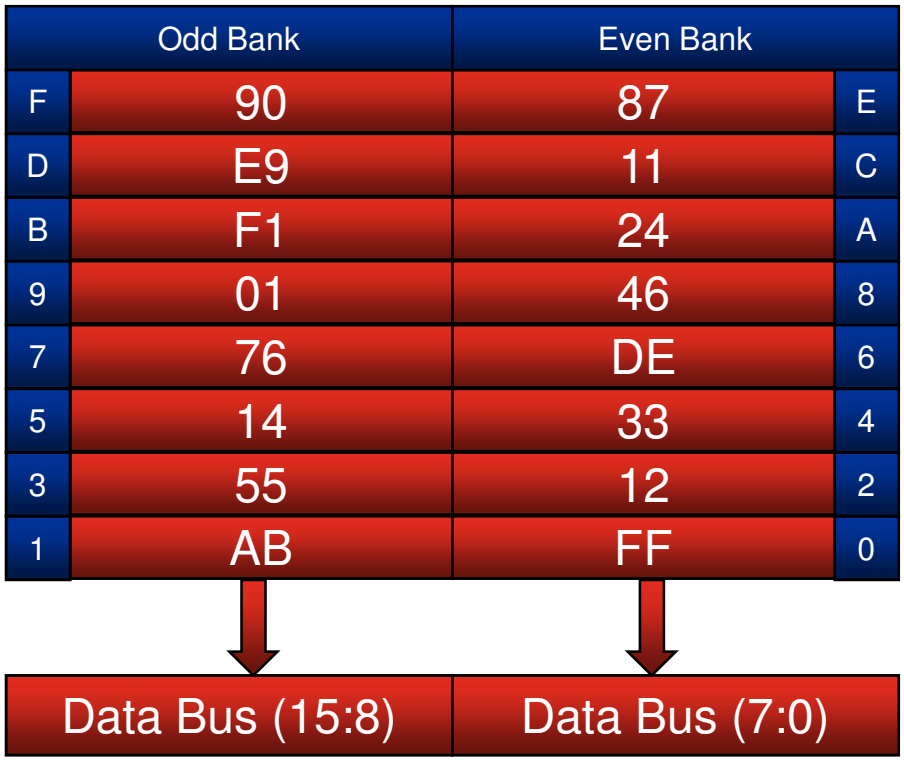
Ordine dei byte nel PIC18F8x20

- Le locazioni 0 and 1 contengono FF e AB…
- ma un accesso a word all'address 0 restituisce ABFF
- I PIC usano l'ordinamento di tipo "little endian" byte order
  - L'indirizzo richiamato (0) punta al byte di peso inferiore del risultato:LSB
  - Il byte di peso maggiore del risultato è preso dal byte di indirizzo sequenziale più alto nella memoria (1): MSB

Byte ordering

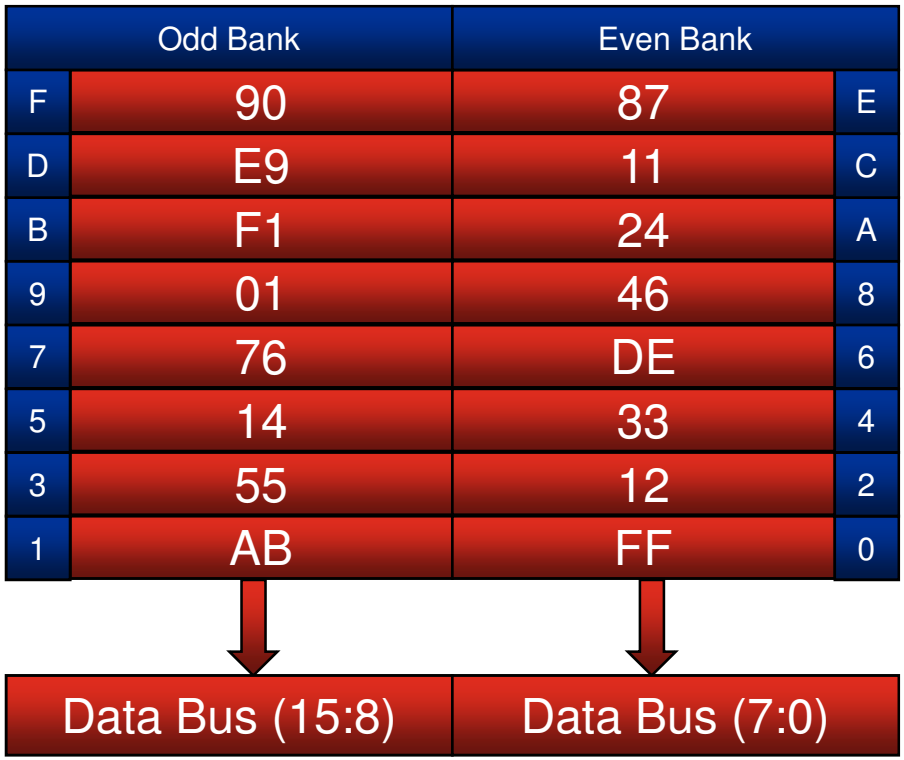
- Little endian vs. big endian
  - Nel caso opposto di codifica Big Endian, il byte di peso maggiore è letto nella locazione di indirizzo richiamata (0), mentre l'LSB è letto nell'indirizzo seguente (1)
  - UNIX lavora in Big Endian, Motorola
- L'ordinamento dei Byte è una caratteristica della architettura.

Allineamento dei Byte



- Leggendo la word all'address 0 data:(15:8)=AB,data(7:0)=FF
- I dati sono passati al data bus nell'ordine corretto di peso dei bit (MSb a sx e LSb a dx)
- Questo è un accesso alla memoria allineato ai pesi dei bit.

Allineamento dei Byte



- Cosa accade se leggo una word all'address 1 ?
- È un accesso valido alla memoria, in linea di principio si può leggere anche a indirizzi dispari, ma quale è lo svantaggio????????
- Il risultato è 12AB
- Ma I byte nel data bus NON SONO ALLINEATI: data(15:8)=AB,data(7:0)=12
- E inoltre devi generare due indirizzi!!! Due accessi al bus

Allineamento dei Byte: Bad/Good habits

- Ogni volta in cui facciamo un accesso alla memoria leggendo più di un byte, i byte letti debbono essere allineati nel corretto ordine in accordo alla architettura
- Se leggiamo una word all'address 1, il byte all'address 2 deve essere memorizzato nell'MSB del data bus e il byte all'address 1 nell'LSB del data bus
- In alcuni micro si era costretti a farlo a mano, in molti micro ora si fa in automatico; nei micro che non sono protetti da questo punto di vista è meglio evitare accuratamente di impantanarsi in questo genere di problemi…….

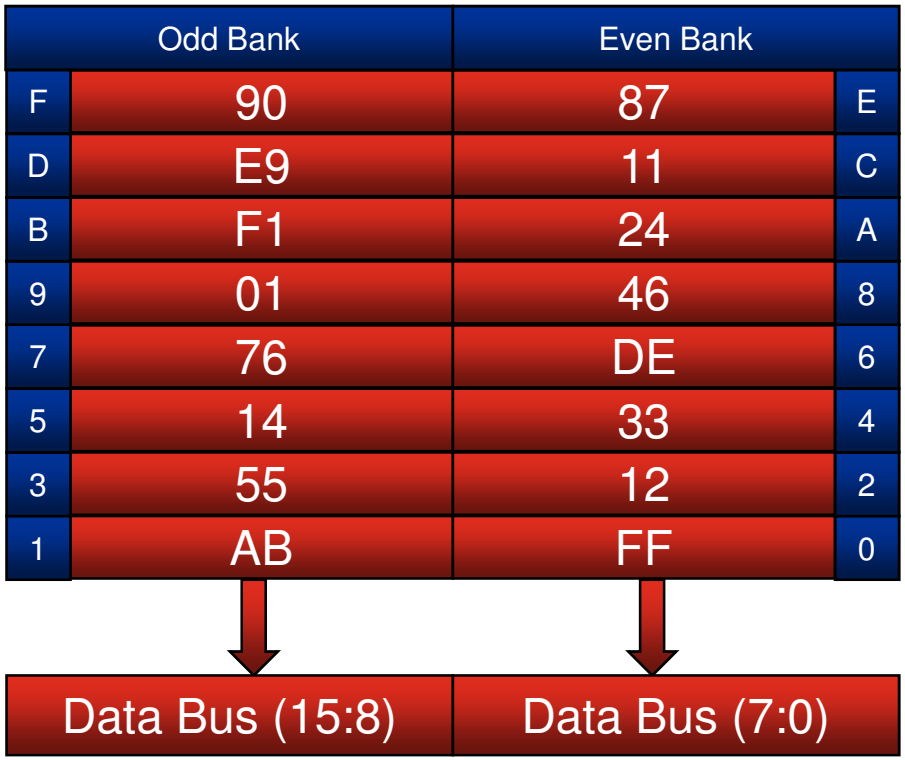
Allineamento dei Byte: Bad/Good habits (2)

- Ci si deve preoccupare di accessi alla memoria non allineati al corretto ordine dei byte ?
- Si, se volete creare un programma che giri veloce e occupi il bus in modo efficiente!
- In accessi non allineati il processore deve:
  - Leggere il byte ad indirizzo dispari (primo accesso alla memoria attraverso il bus)
  - Leggere il byte ad indirizzo pari (secondo accesso alla memoria attraverso il bus)
  - Allineare I due byte (overhead nell'hardware o nel software)
- In un accesso allineato invece il processore deve semplicemente:
  - Leggere il byte pari (un accesso al bus), in automatico si riceve anche il byte dispari se si accede anche alla parte alta del bus dati.
- Accessi allineati sono almeno doppiamente veloci rispetto a quelli non allineati

Tipi di dato

- **Numeri interi**
  - Bit, Nibble, Bytes, Words, Doublewords
  - Signed/unsigned
- **Numeri in floating point**
  - Formati diversi dai numeri interi
- **Testo**
  - 7-bit ASCII char (lettere, caratteri)
  - 8-bit codifica ASCII che permette ulteriori 128 simboli
  - Stringhe: sequenze di char
  - Array: sequenze of numeri, caratteri
- **Files**
  - Immagini (.jpg, .gif, .tiff,…)
  - Video (MPEG, .avi, Quicktime,…)
  - Audio (.wav, .mp3,…)
- Tutto è gestito come sequenza (array) di byte memorizzati
- Il data type dipende dalla modalità di accesso ai dati e dal modo di usarli

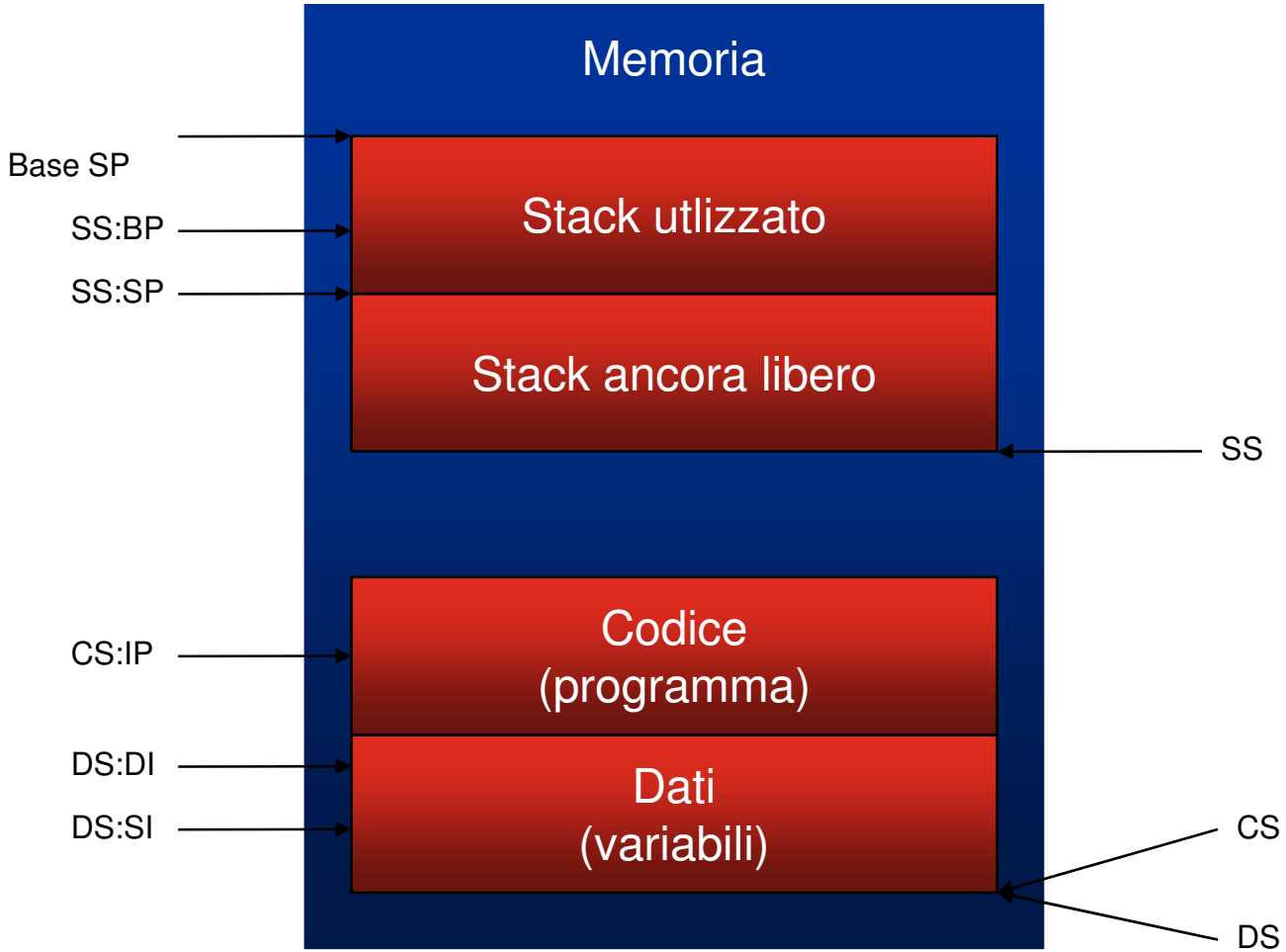
Tipi di Dato



- 16 signed/unsigned 8-bit byte
- 8 signed/unsigned 16-bit word
- 4 signed/unsigned 32-bit doubleword
- Una stringa
- Istruzioni e operandi (se program memory)…

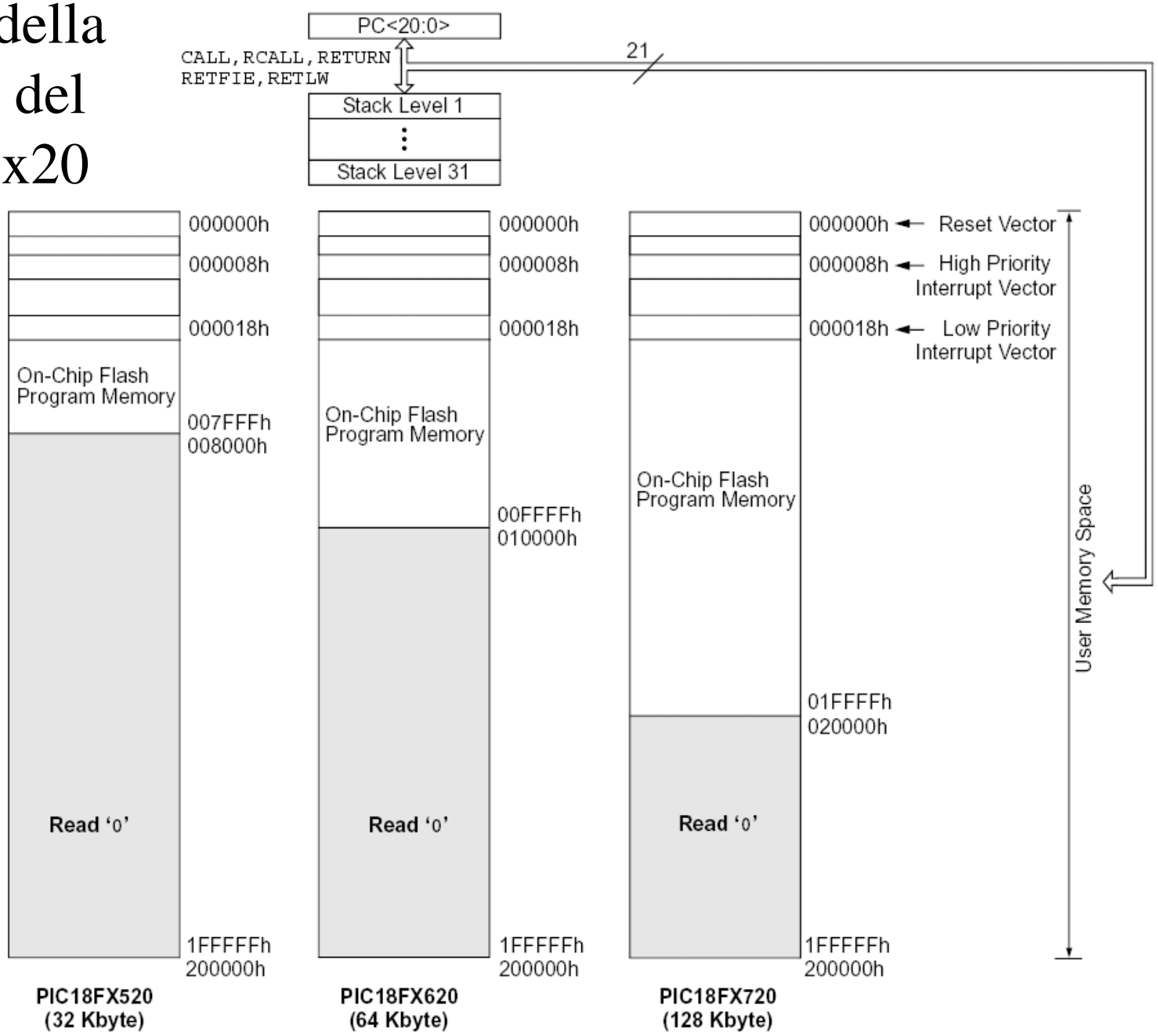
Memoria segmentata

- Partizione logica di un programma



Struttura della memoria del PIC18F8x20

FLASH



Struttura della memoria del PIC18F8x20

R
A
M