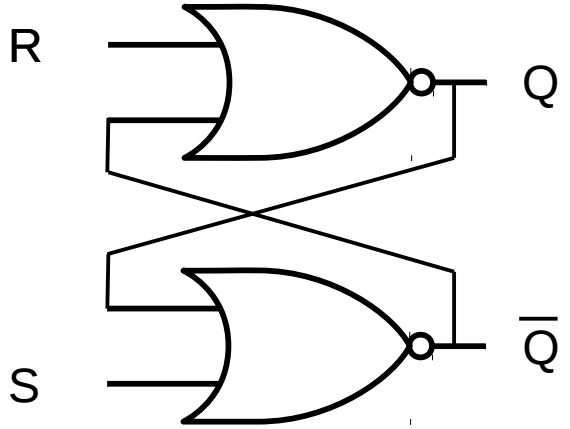
Cenni di architetture dei calcolatori

E Mumolo

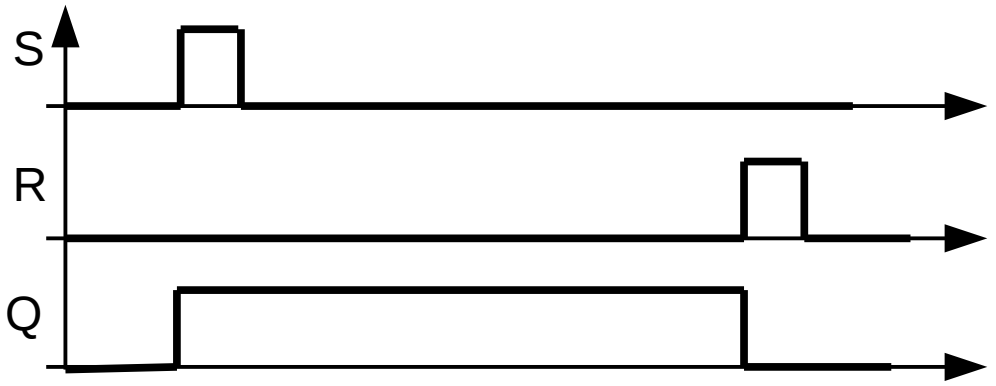
Memoria

- Memoria da 1 bit (latch SR)



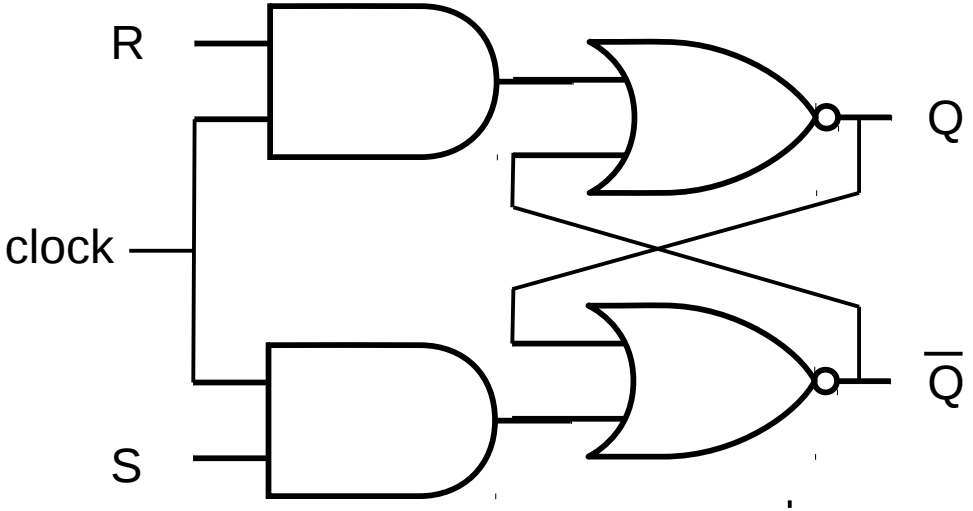
S	R	Q_{n+1}	Q_{n+1}
0	0	Q_n	Q_n
0	1	0	1
1	0	1	0
1	1	0	0 ← Incongruente, da evitare

- Timing



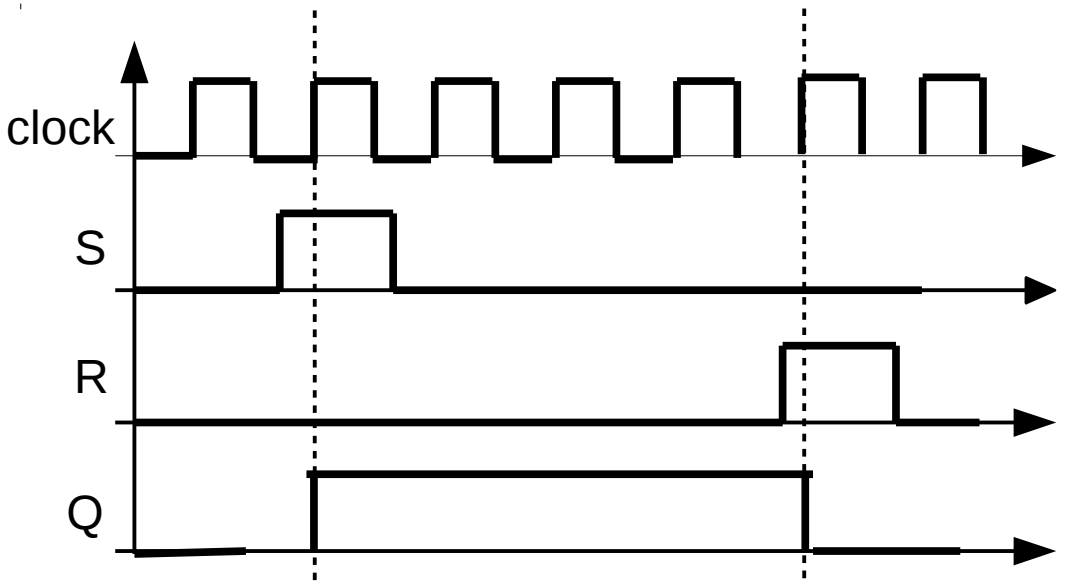
Memoria

- Clocked latch SR



S	R	Q_{n+1}	Q_{n+1}
0	0	Q_n	Q_n
0	1	0	1
1	0	1	0
1	1	0	0

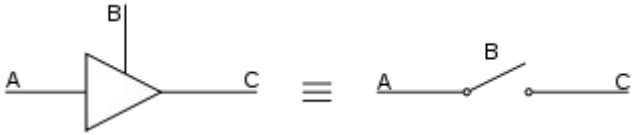
- Timing

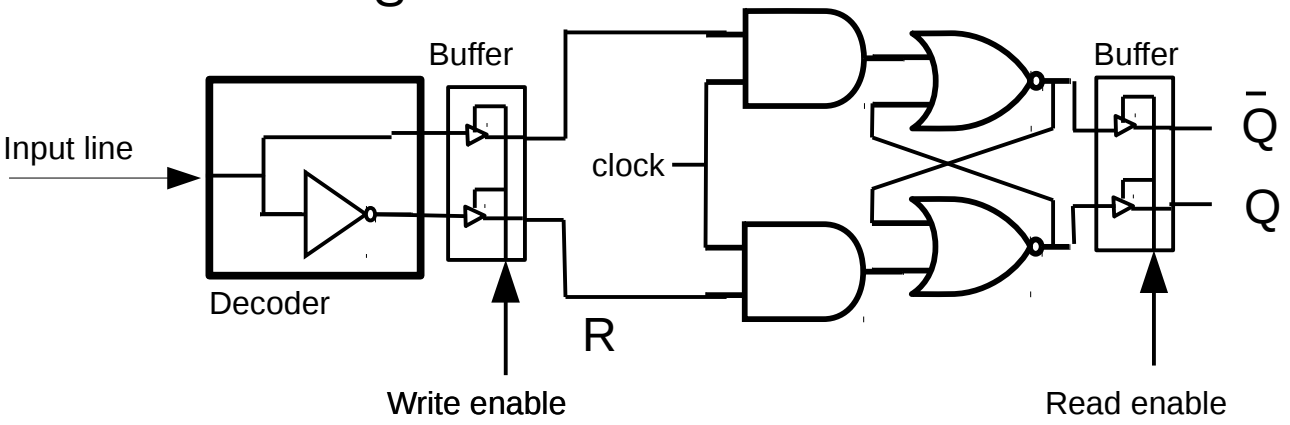


Memoria

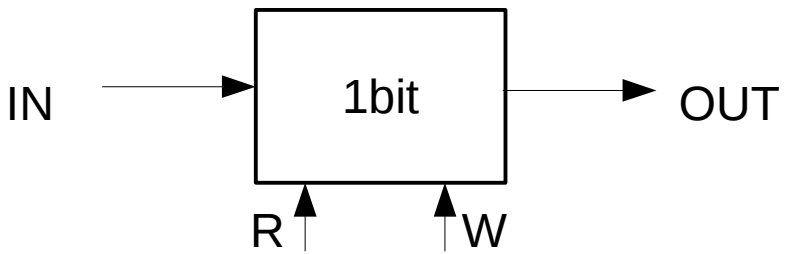
- Cella di memoria da 1 bit

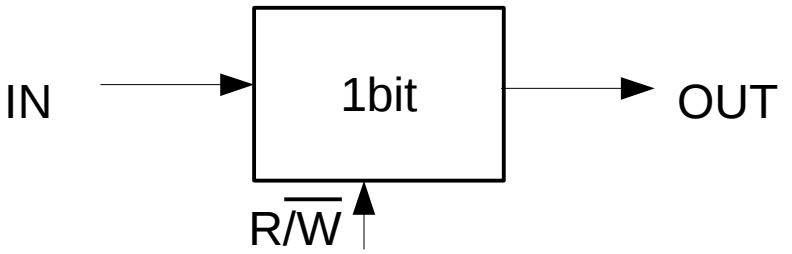
Buffer tristate:





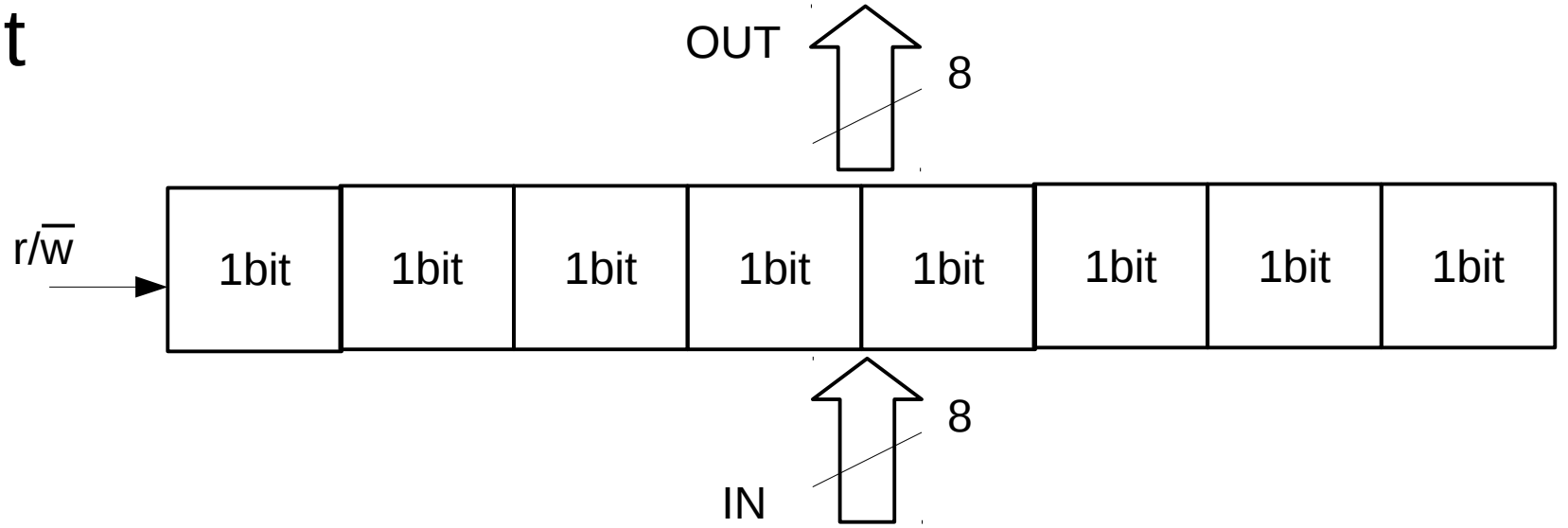
- Cella di memoria da 1 bit



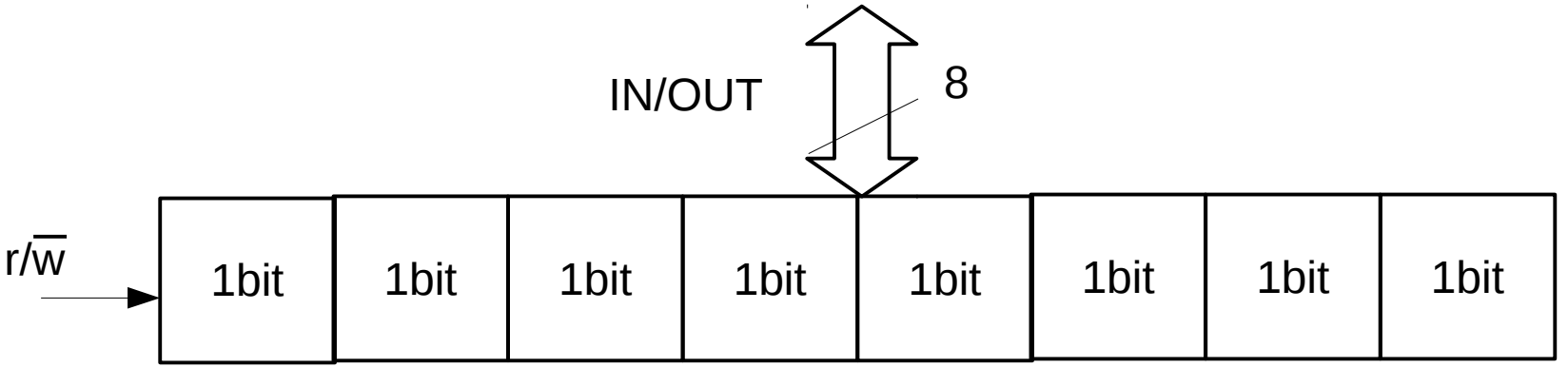


Memoria

- Array di Celle di memoria da 8 bit = registro da 8 bit

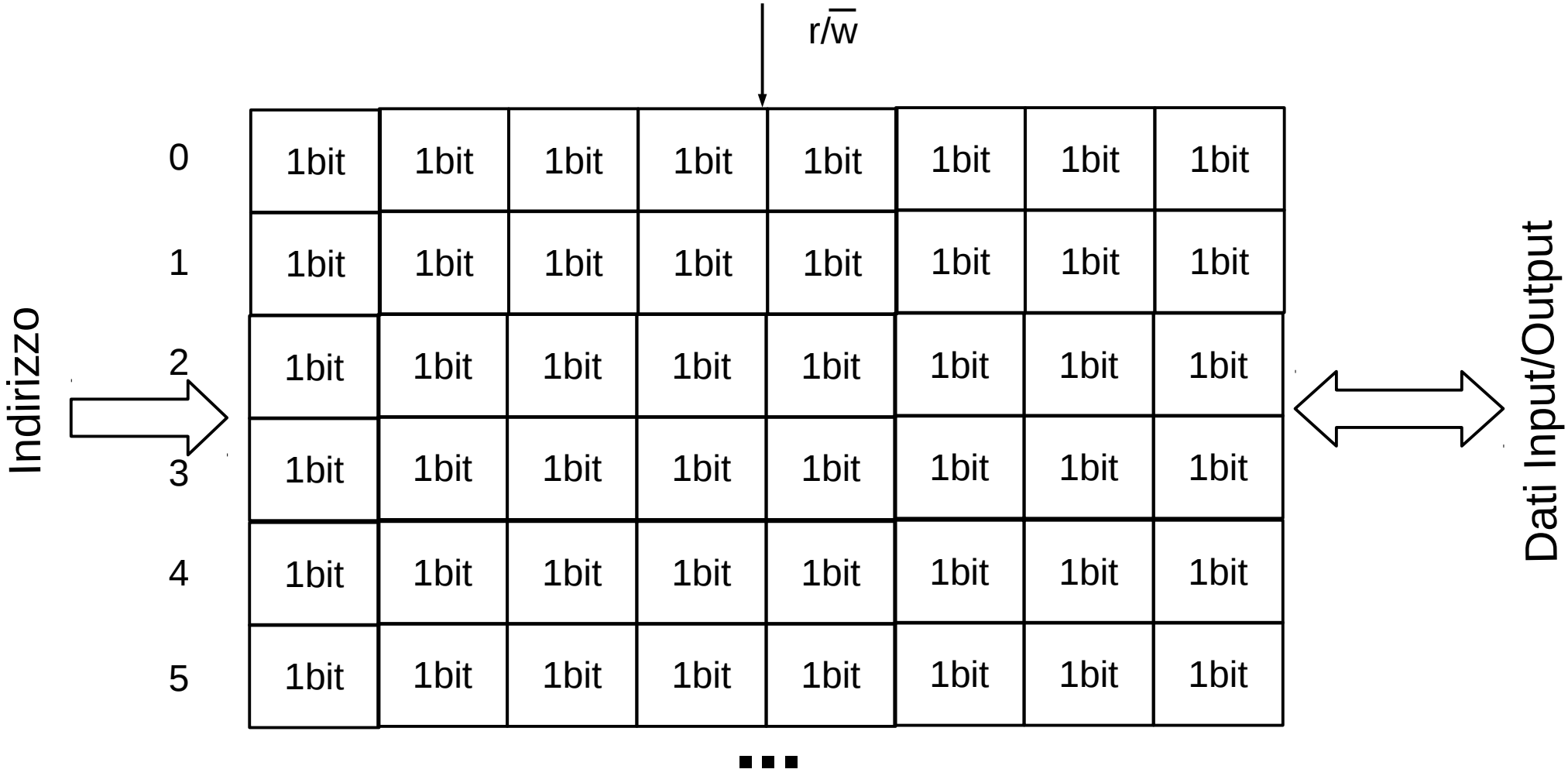


- Normalmente:



Memoria

- Array di Celle di memoria da 8 bit = Memoria

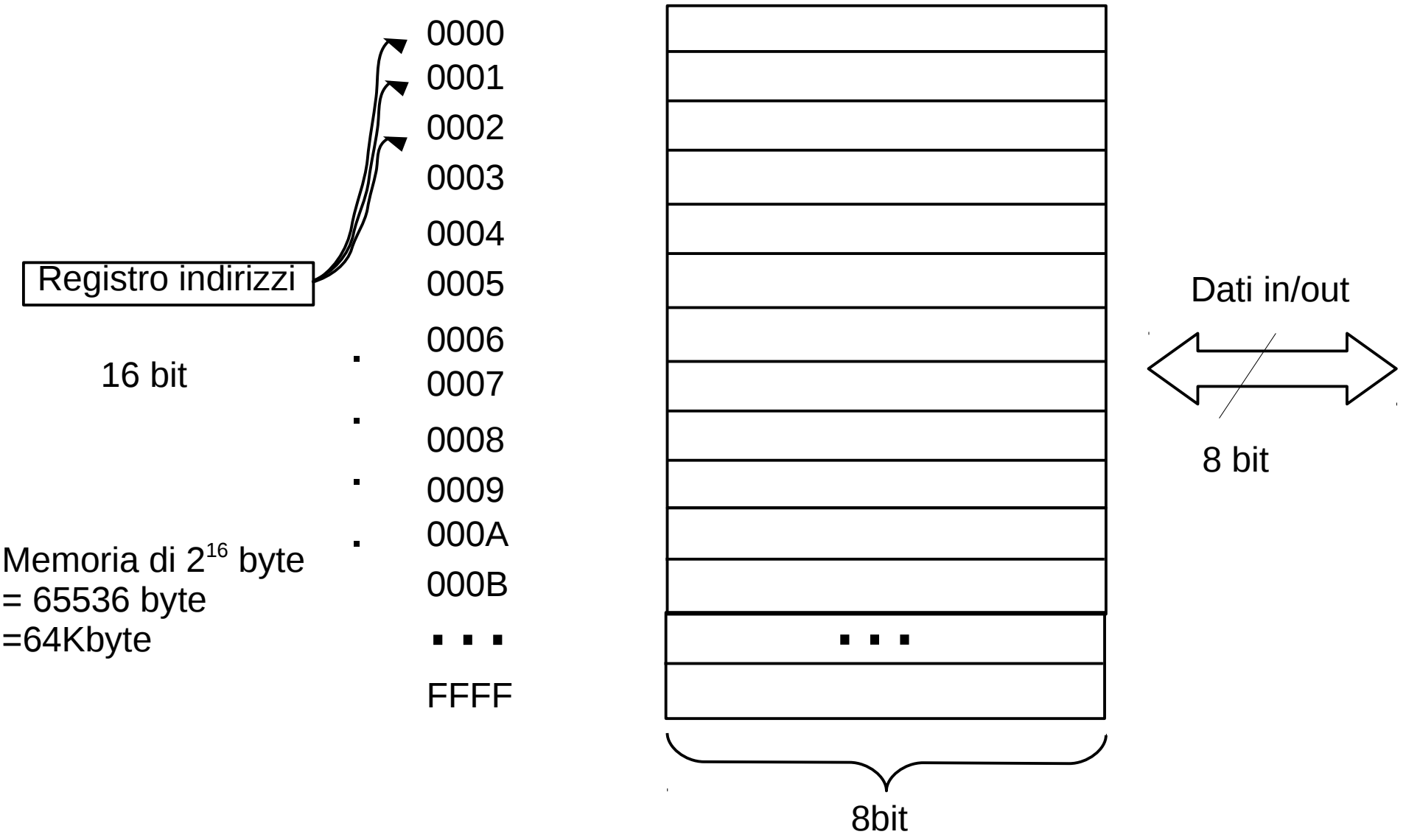


Memoria

- Strutture fisiche standard
  - Bit informazione elementare 0/1
  - Nibble gruppo di 4 bit
  - Byte gruppo di 2 nibble (8 bit)
  - Word gruppo di 2 byte (16 bit)
  - Dword gruppo di 2 word (32 bit)
  - Qword gruppo di 2 dword (64 bit)

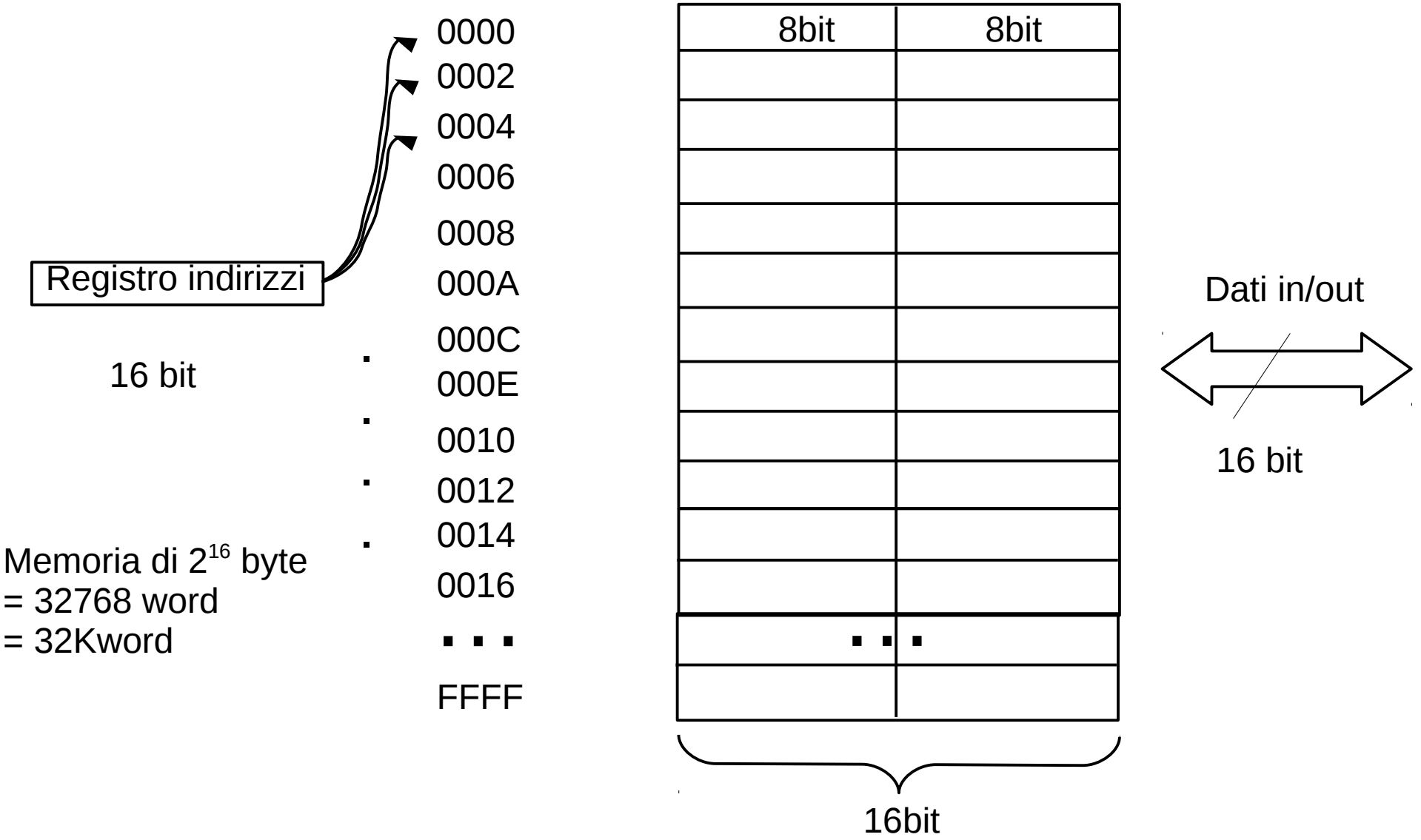
Memoria

- Organizzazione della memoria per byte



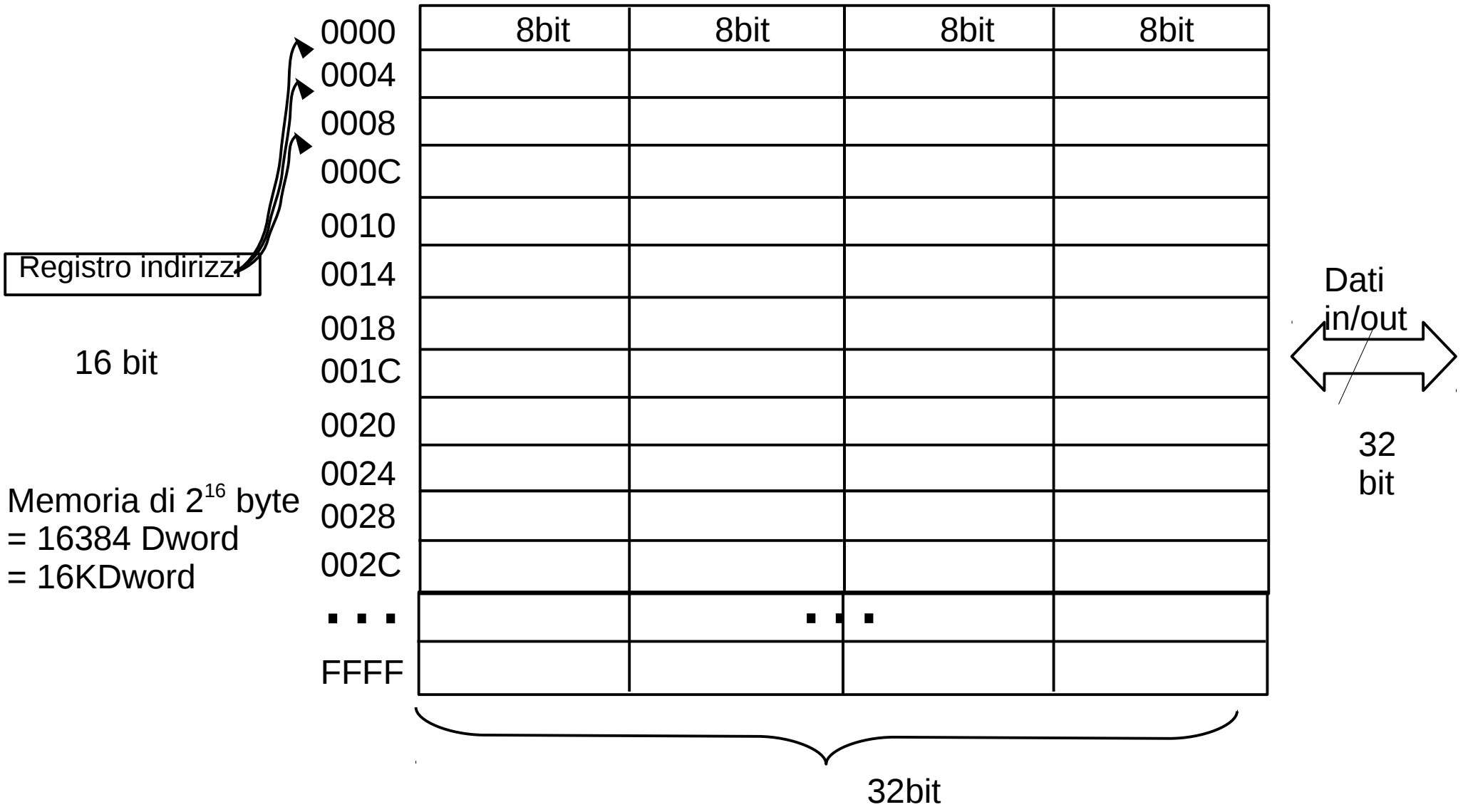
Memoria

- Organizzazione della memoria per word



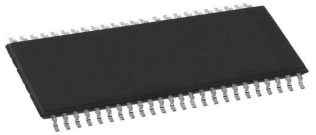
Memoria

- Organizzazione della memoria per Dword



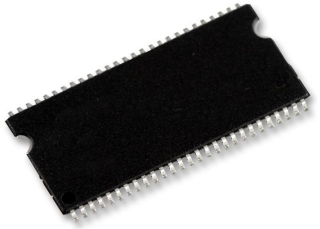
Memoria

- SRAM: Static RAM



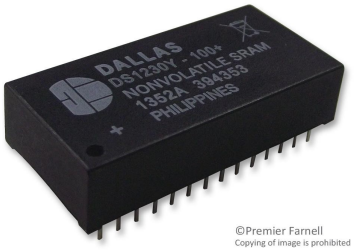
Tempo di Accesso: 7ns
Configurazione Memoria SRAM: 32M x 8bit
Prezzo 19euro

- DRAM: Dynamic RAM



Tempo di Accesso: 15ns
Configurazione Memoria DRAM: 32M x 8bit
Prezzo 7euro

- Non Volatile con batteria di backup



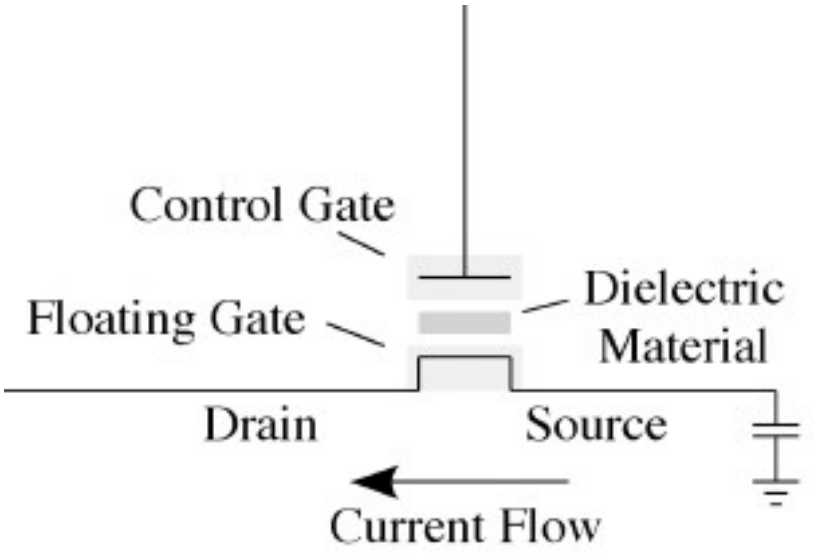
Tempo di Accesso: 100ns
Configurazione Memoria NVRAM: 32K x 8bit
Prezzo 28euro
Lithium backup battery

Memoria non volatile

- FRAM=Ferroelectric RAM: uno strato ferroso tra due strati di dielettrico
- MRAM=Magnetoresistive RAM: elementi magnetici tra due strati di dielettrico
- PRAM=Phase-change RAM: utilizza un particolare tipo di vetro che cambia stato con un raggio laser
- RRAM=Resistive RAM: utilizza il memristor: dispositivo la cui resistenza dipende dalla quantità di carica elettrica passata in precedenza.
- FLASH: memorizza le informazioni sotto forma di carica elettrica contenuta in materiale isolato

Memoria non volatile

- Struttura di una cella della memoria Flash



- L'informazione 0/1 corrisponde alla carica immagazzinata nel dielettrico

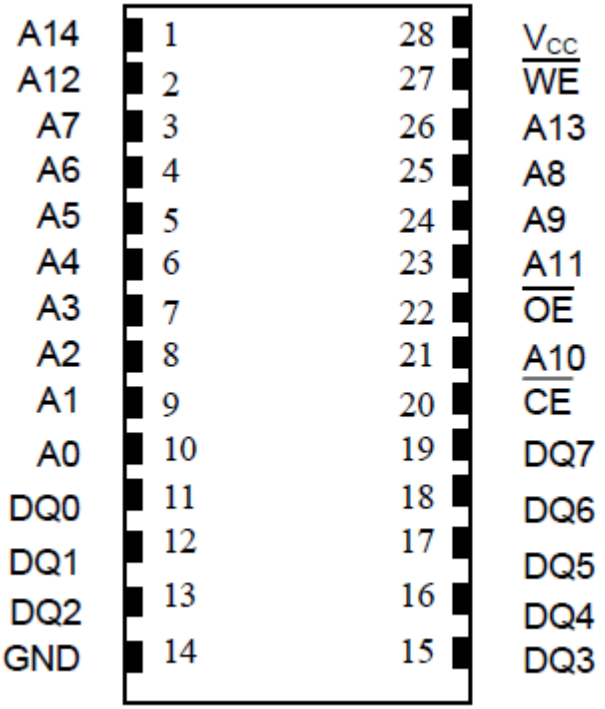
Memoria a sola lettura

- UVEPROM=Ultra-Violet Erasable Programmable Read Only Memory: programmati elettronicamente, cancelati con raggi ultravioletti

- EEPROM=Electrically Erasable Programmable Read-Only Memory
- ROM=Read Only Memory
- PROM=Programmable Read-Only Memory

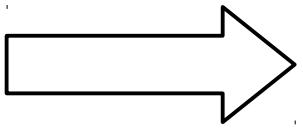
Esempio: memoria DS1230 della Maxim

- 32k x 8 bit
- A0-A14: bus indirizzi
- DQ0-DQ7: dati IN/OUT
- CE: chip enable
- WE: write enable
- OE: output enable
- VCC: +5V
- GND: massa



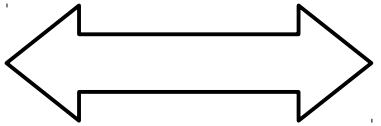
BUS

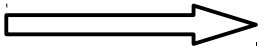
- Comunicazione elettrica tra dispositivi.
  - Tipi di bus:



Bus indirizzi

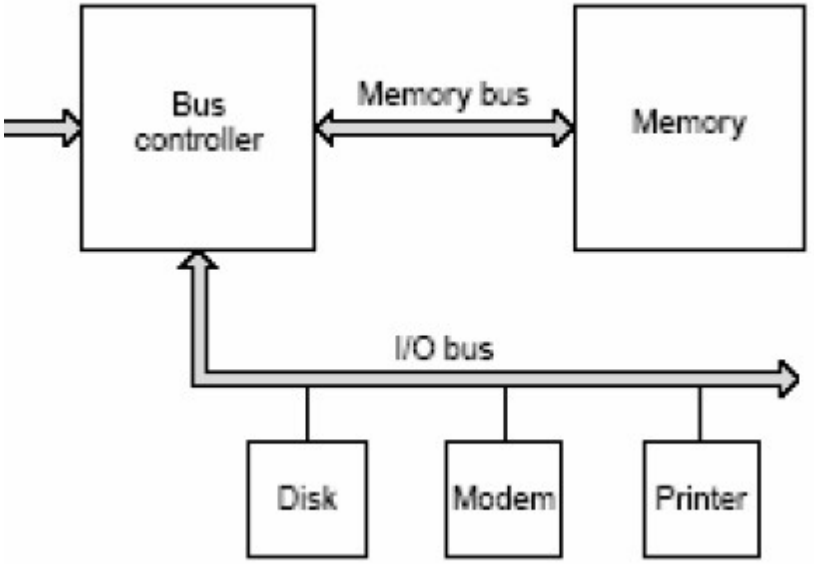
Bus dati bidirezionale





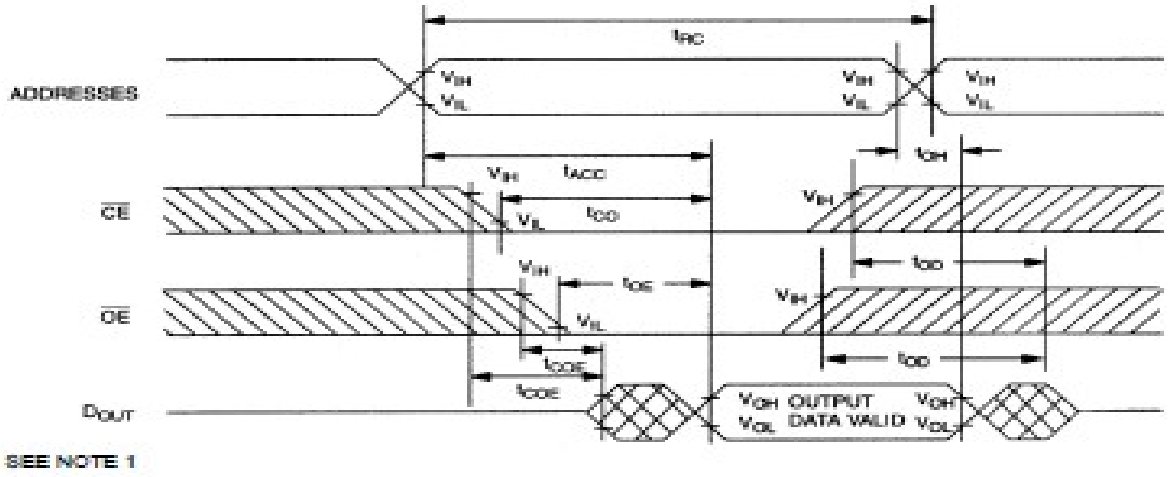
Bus controllo

- Esempio di collegamento (memory bus si riferisce a indirizzi, dati, controllo)

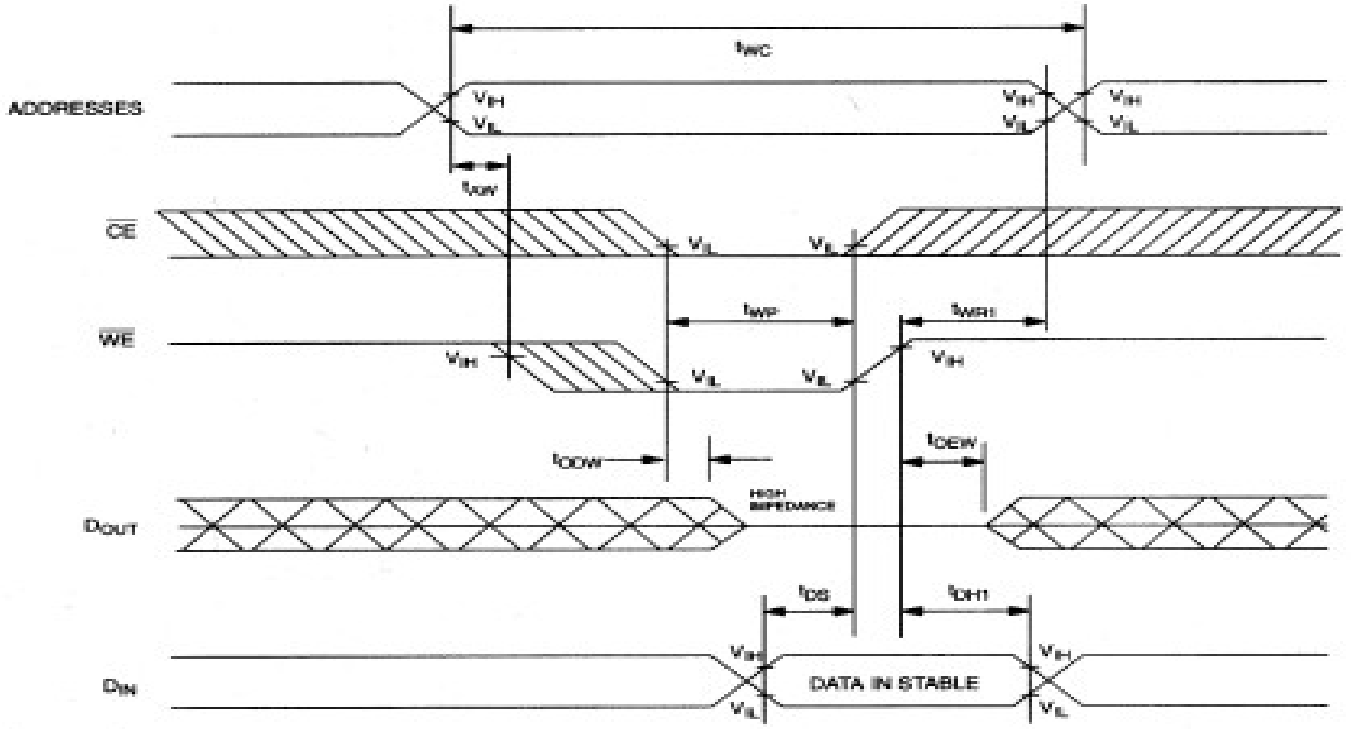


Segnali su BUS memoria DS1230

READ CYCLE



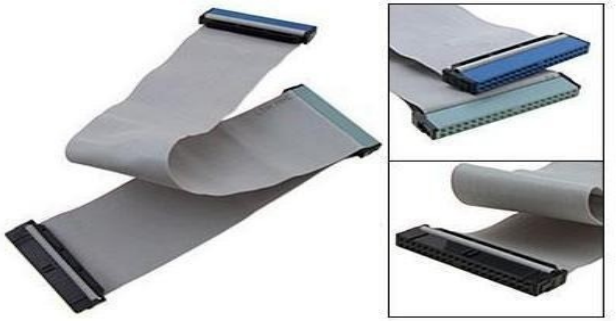
WRITE CYCLE 1



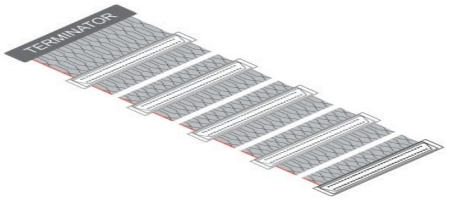
Tipi di BUS esterni

- SATA (Serial Advanced Technology Attachmen), Bus computer e standard di interconnessione via cavo

- IDE (Integrated Drive Electronics), è uno standard per le interfacce utilizzato per i dispositivi che si collegano alla scheda madre



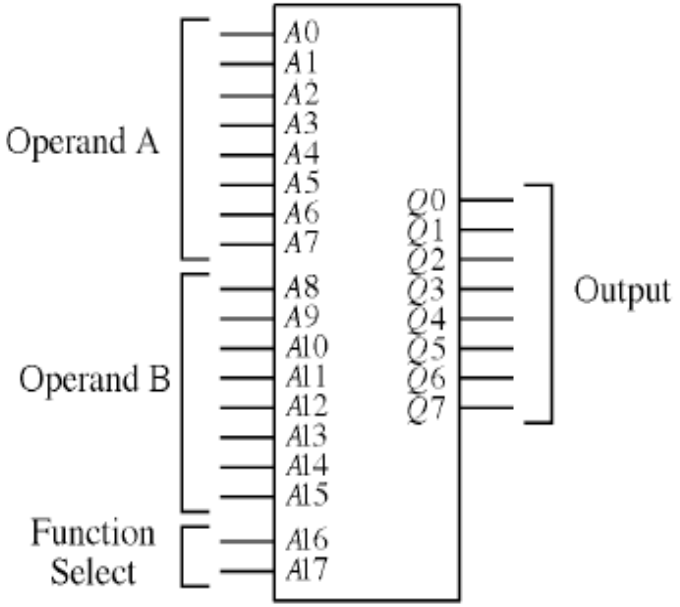
- SCSI (Small Computer System Interface), interfaccia standard progettata per realizzare il trasferimento di dati a bus in modalità parallela.



- Bus su PCB

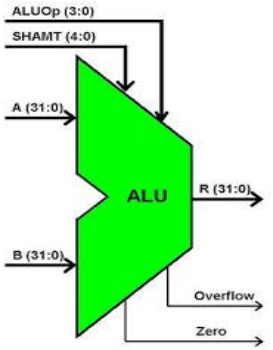
ALU

- Modulo digitale che realizza varie operazioni aritmetiche e logiche
- Rappresentazione funzionale:



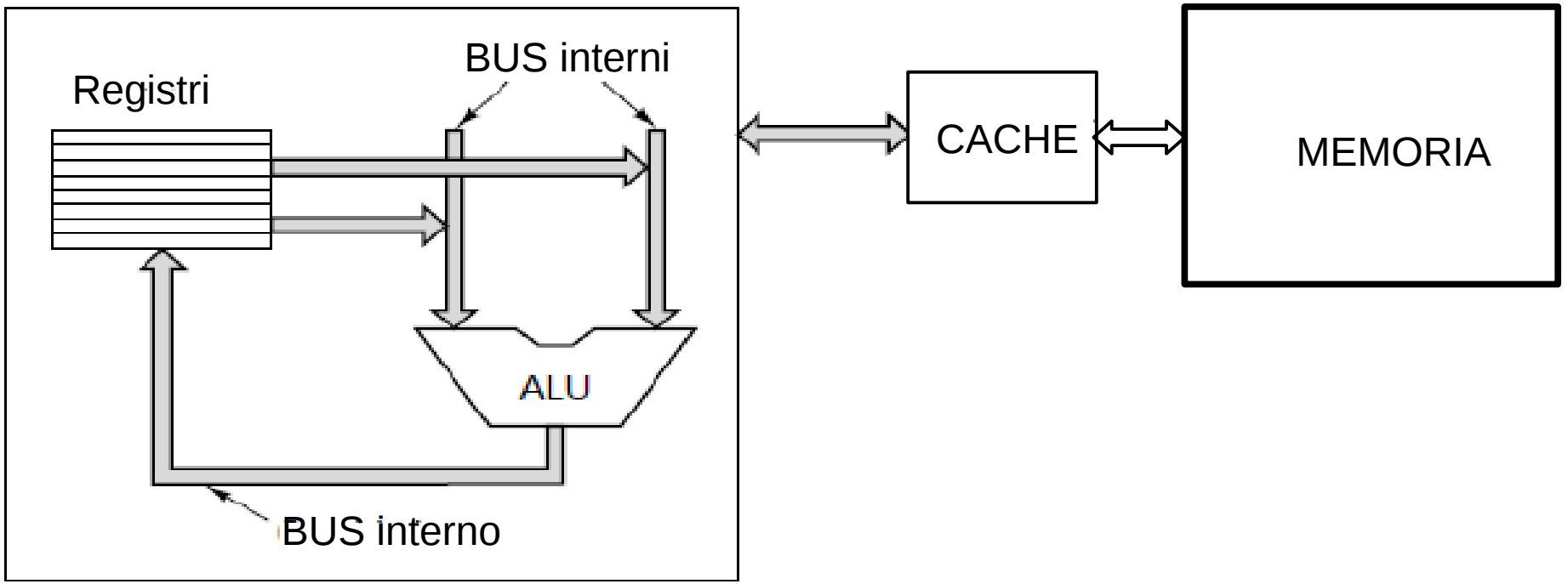
A17	A16	Function
0	0	Add
0	1	Subtract
1	0	Multiply
1	1	Divide

- Simbolo standard:

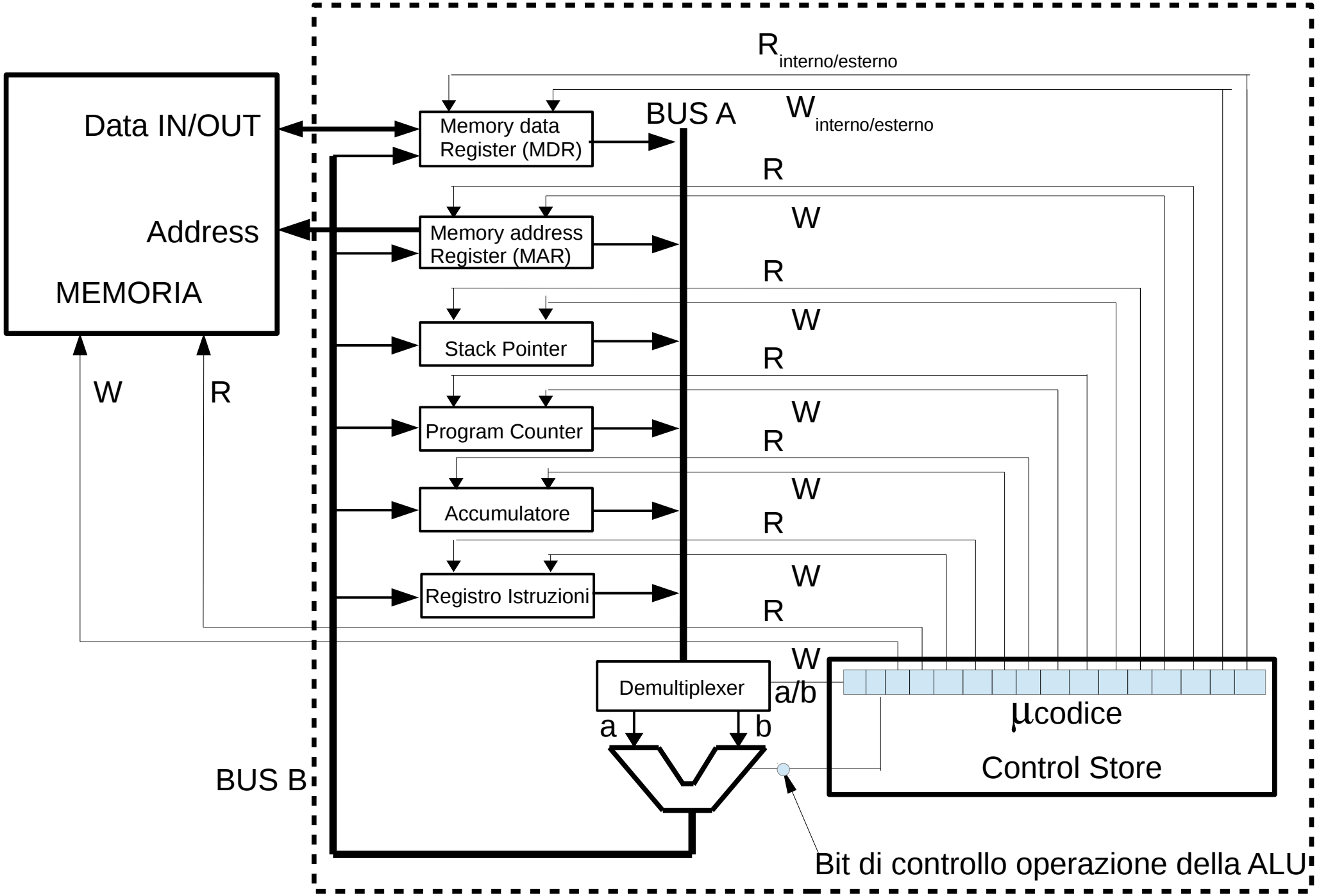


CPU

- Mettiamo insieme i componenti visti finora in uno schema di massima



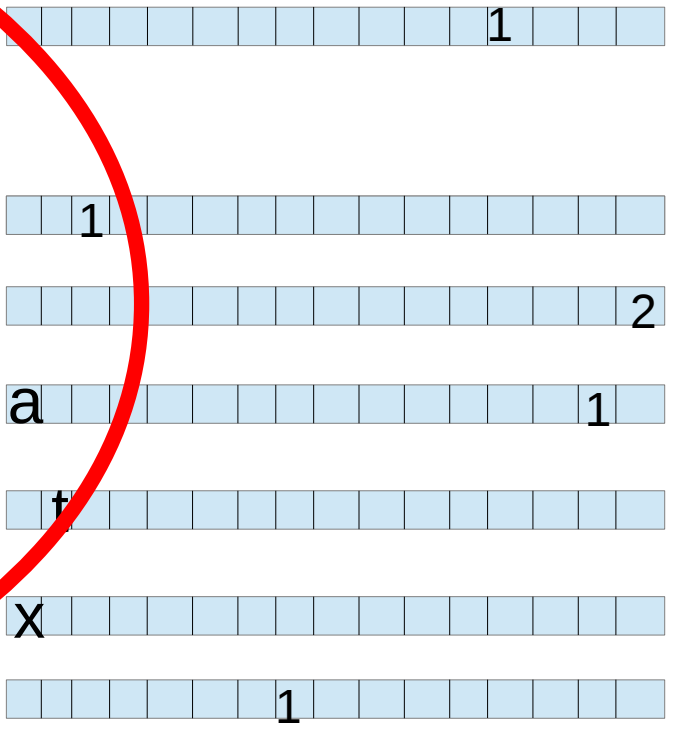
CPU: Schema di massima un po' piu' dettagliato



CPU: operazioni sui dati

- Una prima operazione d'esempio: caricare in Accumulatore il contenuto di una cella di memoria di indirizzo dato
  - MAR=<indirizzo>
  - Bit scrittura W di MAR
  - Attesa
  - Bit W della MEMORIA esterna
  - Bit $R_{esterno}$ di MDR
  - Bit $W_{interno}$ di MDR
  - Bit 'a' di Demux
  - Bit di controllo ALU → modalita' trasparente
  - Bit lettura R di Accumulatore

LDA <address>
oppure
MOVE <address>,Dn



- Questo programma (microcodice) e' memorizzato nella Control Store

CPU: operazioni sui dati

- Un'altra operazione d'esempio: caricare il contenuto di Accumulatore in una cella di memoria di indirizzo dato

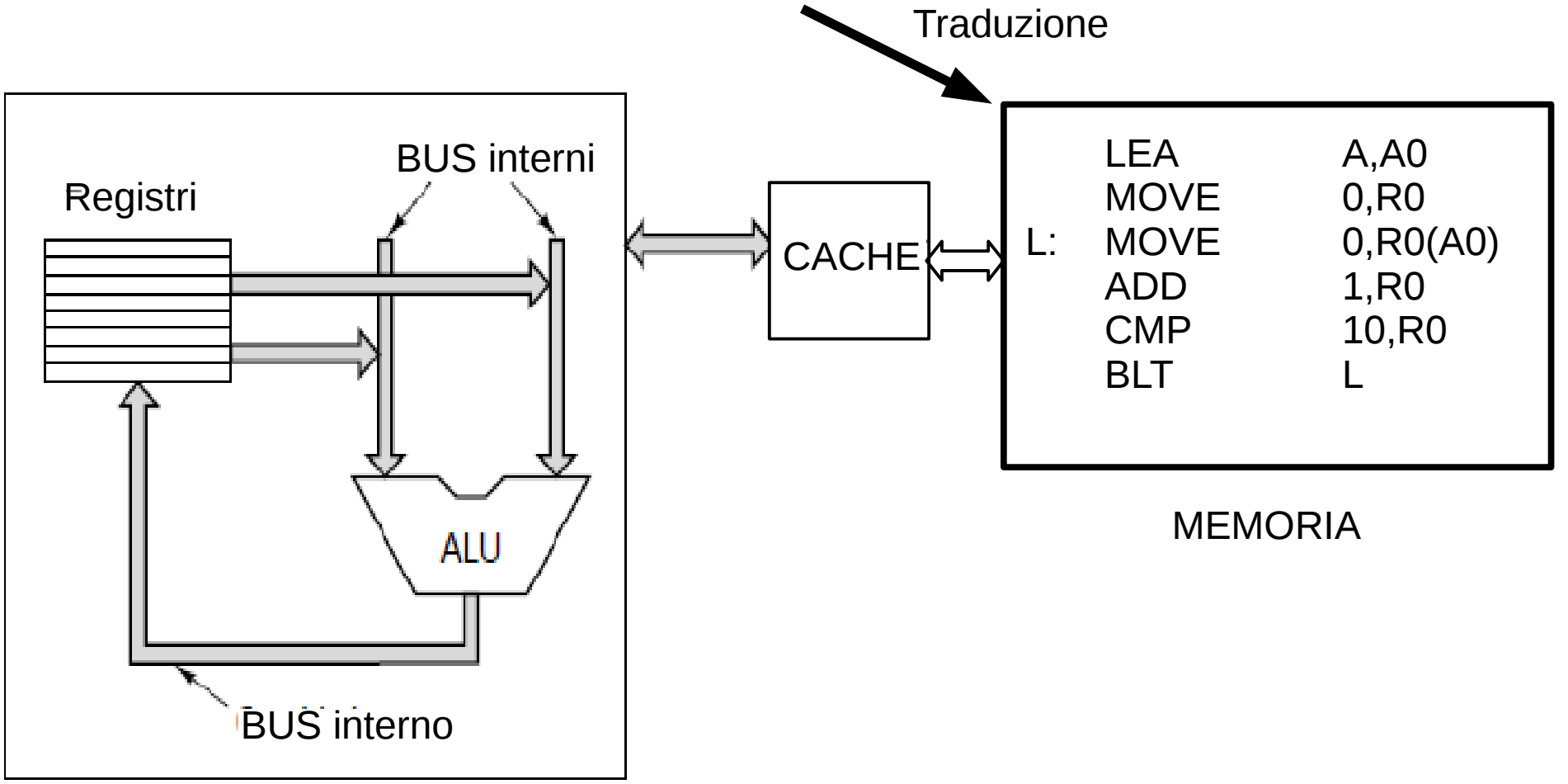
STA <address>
oppure
MOVE Dn,<address>

  - Bit scrittura W di Accumulatore
  - Bit di controllo Demux su 'a'
  - Parola di controllo ALU → modalita' trasparente
  - Bit $R_{interno}$ di MDR
  - Carica MAR=<address>
  - Bit W di MAR (scrive l'indirizzo sul bus della memoria)
  - Bit $W_{esterno}$ di MDR (scrive il dato sul bus della memoria)
  - Bit R della MEMORIA

- Anche questo programma (microcodice) e' memorizzato nella Control Store

Livello MACCHINA → Istruzioni Assembler

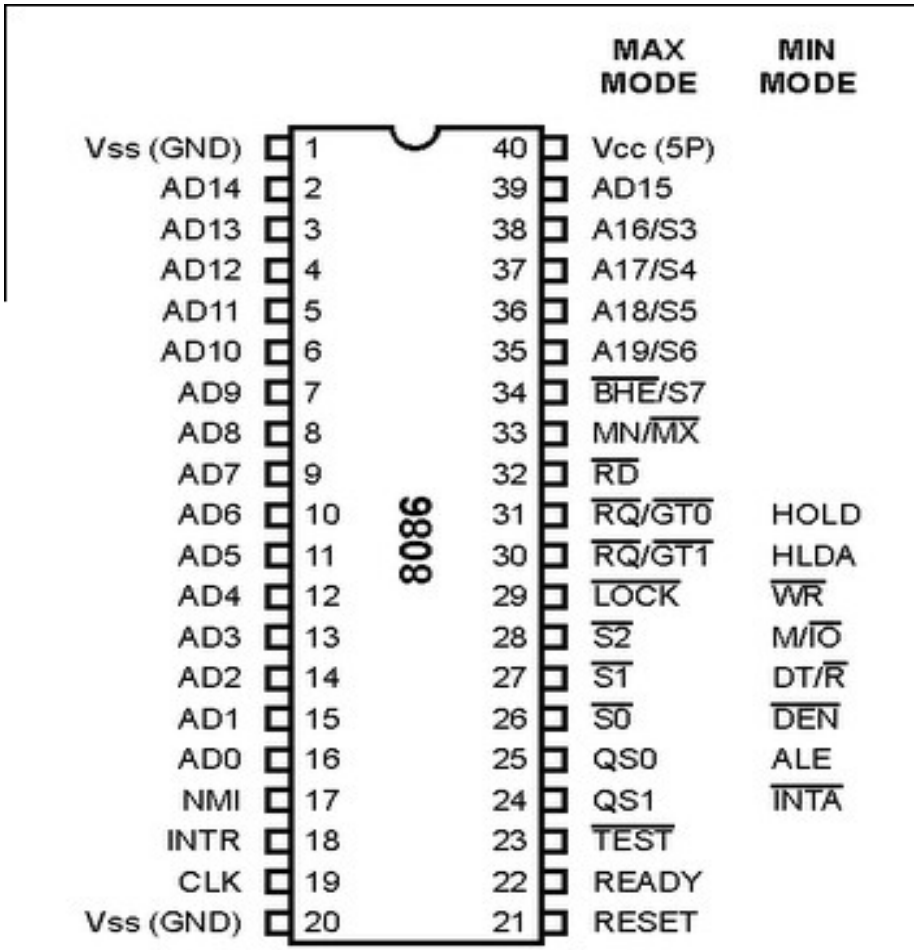
- Il programma e' espresso in istruzioni Assembler che vengono eseguite dal microcodice della CPU
- Esempio: `for(i=0;i<10;i++) A[i]=0`



Interrupt hardware

- E' un segnale asincrono: si è verificato un evento
- L'evento deve essere gestito
- Sorgenti dell'interrupt hardware: periferiche
- Gestione interruzione: “Exception Processing”
- L'evento si gestisce con un sottoprogramma
- Exception Processing è il meccanismo di chiamata del sottoprogramma

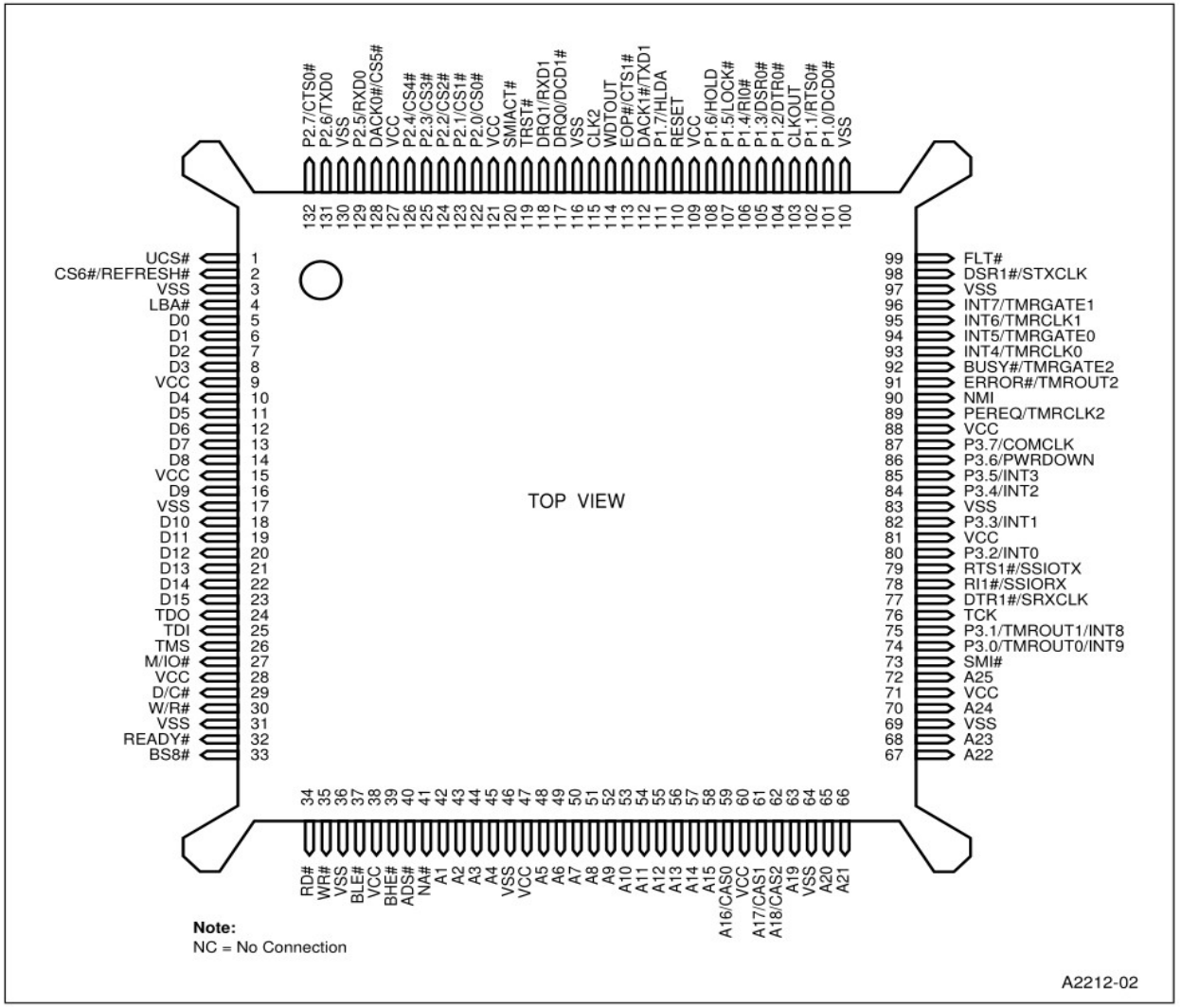
Linee di Interrupt hardware in 8086



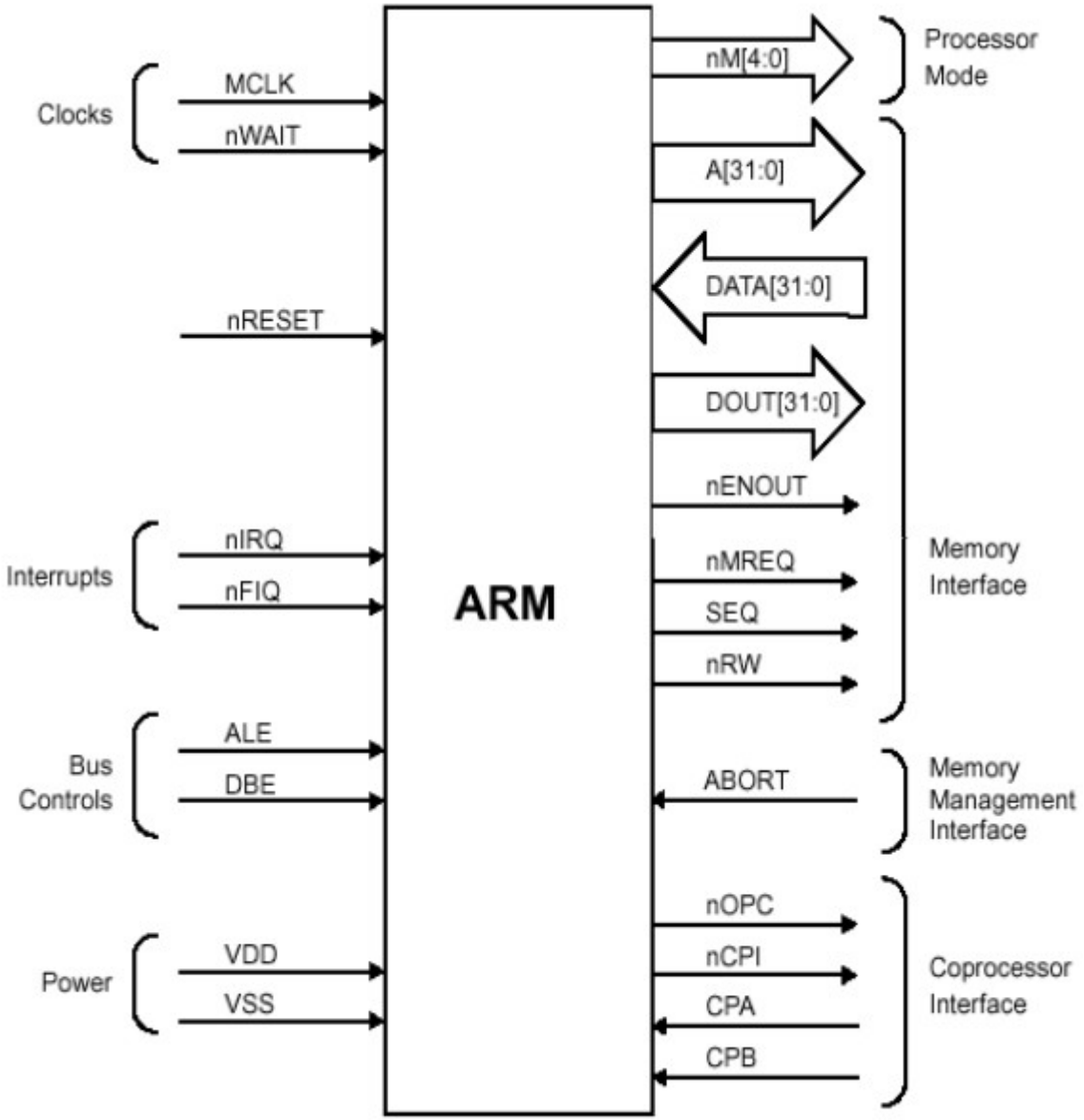
Linee di Interrupt nel 386

2.0 Pin Assignment

Figure 2. Intel386™ EX Embedded Processor 132-Pin PQFP Pin Assignment



Linee di Interrupt in ARM

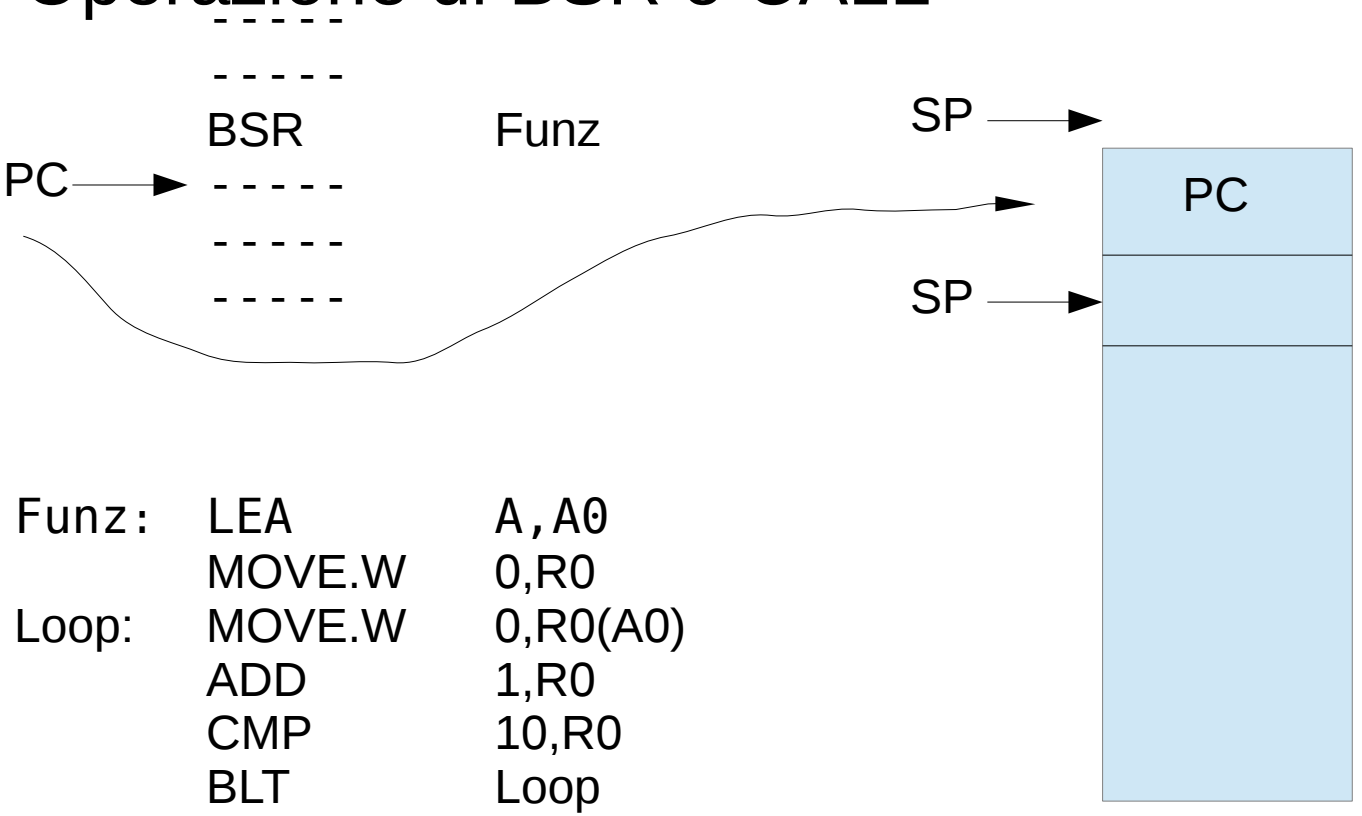


Memoria Stack

- STACK è una regione di memoria utilizzata in modalità LIFO (last in first out)
- E' una memoria dinamica usata normalmente dall'alto verso il basso
- Moltissimi utilizzi:
  - Da algoritmi particolari
  - Dai moduli software per memorizare variabili locali, valori intermedi, parametri di funzioni, valori di ritorno, gestione eccezioni, dalla CPU,...
- La CPU utilizza il registro Stack Pointer (SP) per indirizzare la memoria Stack

Memoria Stack

- Torniamo all'esempio del livello macchina
- Operazione di BSR o CALL

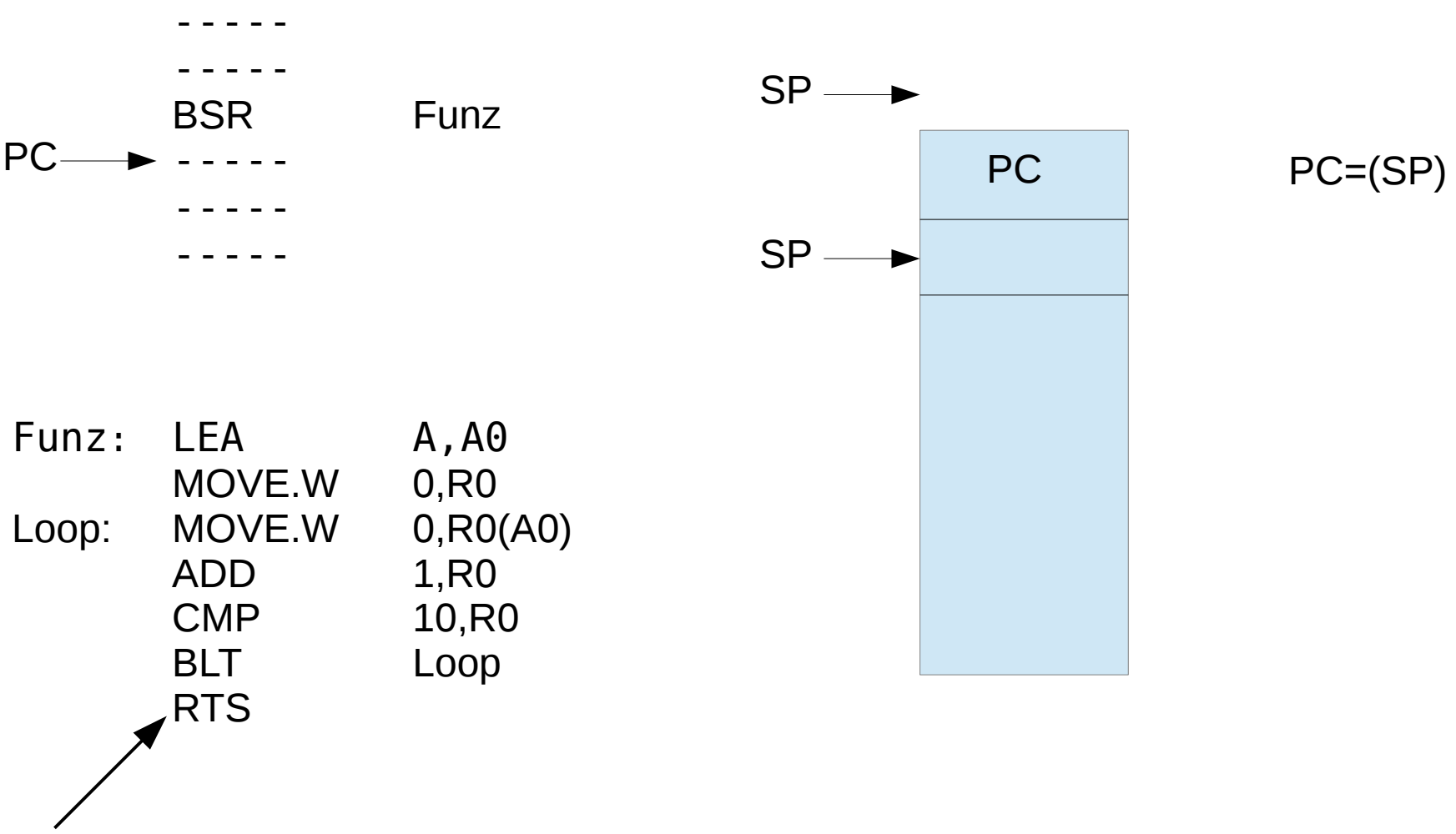



```
Funz:   LEA       A,A0
        MOVE.W    0,R0
Loop:   MOVE.W    0,R0(A0)
        ADD       1,R0
        CMP       10,R0
        BLT       Loop
        RTS
```

Esempio in freescale coldfire. In X86 si usa CALL/RET

Memoria Stack

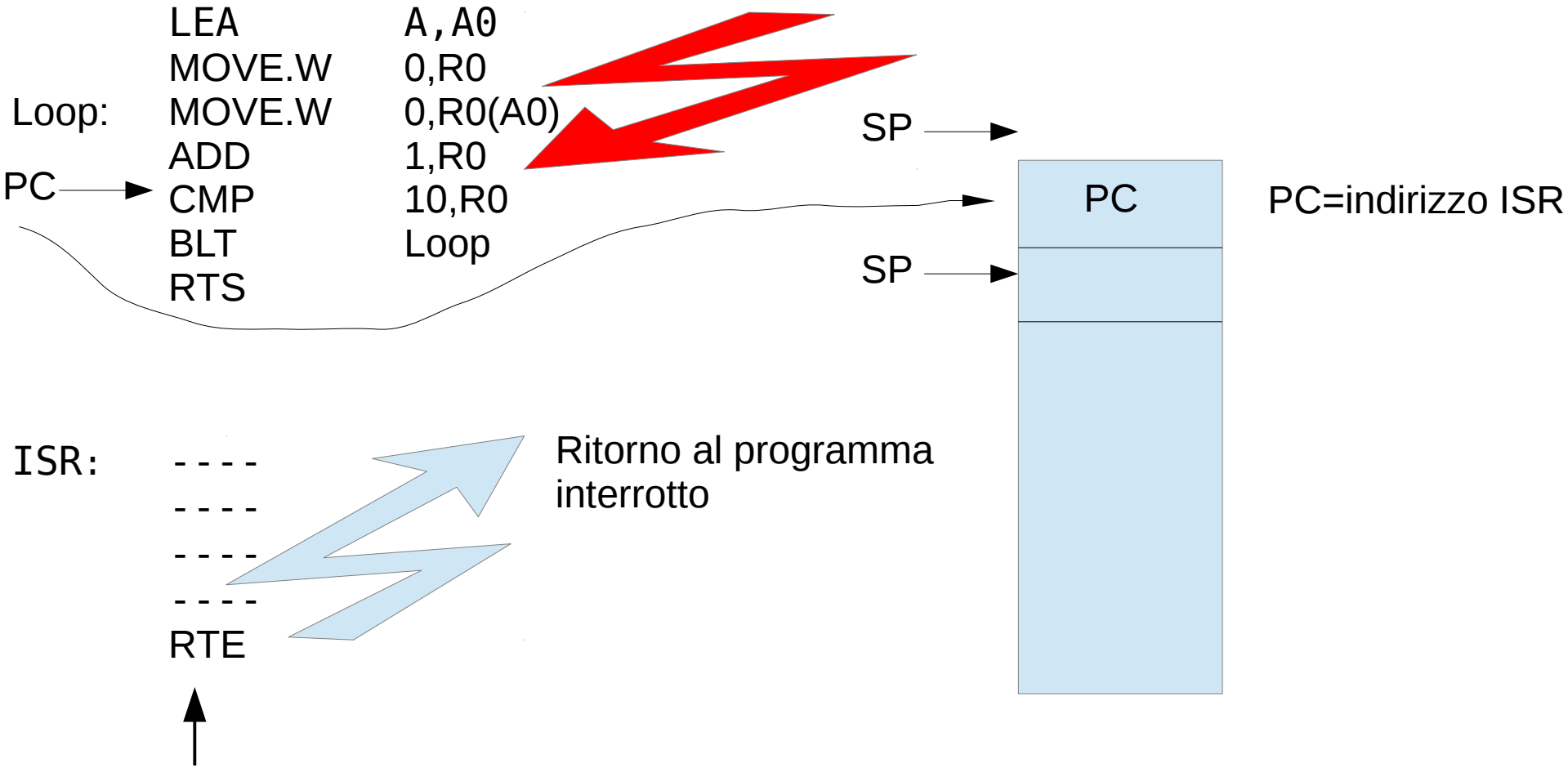
- Operazione di RTS o RET



Esempio in freescale coldfire. In X86 si usa CALL/RET

Exception Processing

- Supponiamo scatti un interrupt hardware



Ritorno da eccezione: preleva dallo stack l'indirizzo di ritorno

- Gli interrupt software vengono gestiti allo stesso modo!

Sorgenti di interrupt

- Segnali esterni (interrupt hardware). Esempi:
  - Timer
  - Arrivo di un carattere sulla seriale
  - Un ADC ha finito la conversione
- **Istruzioni (interrupt software)**
  - **Vari tipi di istruzioni TRAP <numero>**
- Microcodice
  - Operazioni aritmetiche (divisione per zero)
  - Istruzioni illegali
  - Errori di BUS
  - Errori di indirizzo

Cosa viene interrotto dall'interrupt?

- Codice e microcodice

  Esempio: microcodice di STA <address>:

  - Bit di controllo Demux su 'a'
  - Parola di controllo ALU → modalita' trasparente
  - Bit $R_{interno}$ di MDR
  - Carica MAR=<address>
  - Bit W di MAR (scrive l'indirizzo sul bus della memoria)
  - Bit $W_{esterno}$ di MDR (scrive il dato sul bus della memoria)
  - Bit R della MEMORIA
  - Bit scrittura W di Accumulatore

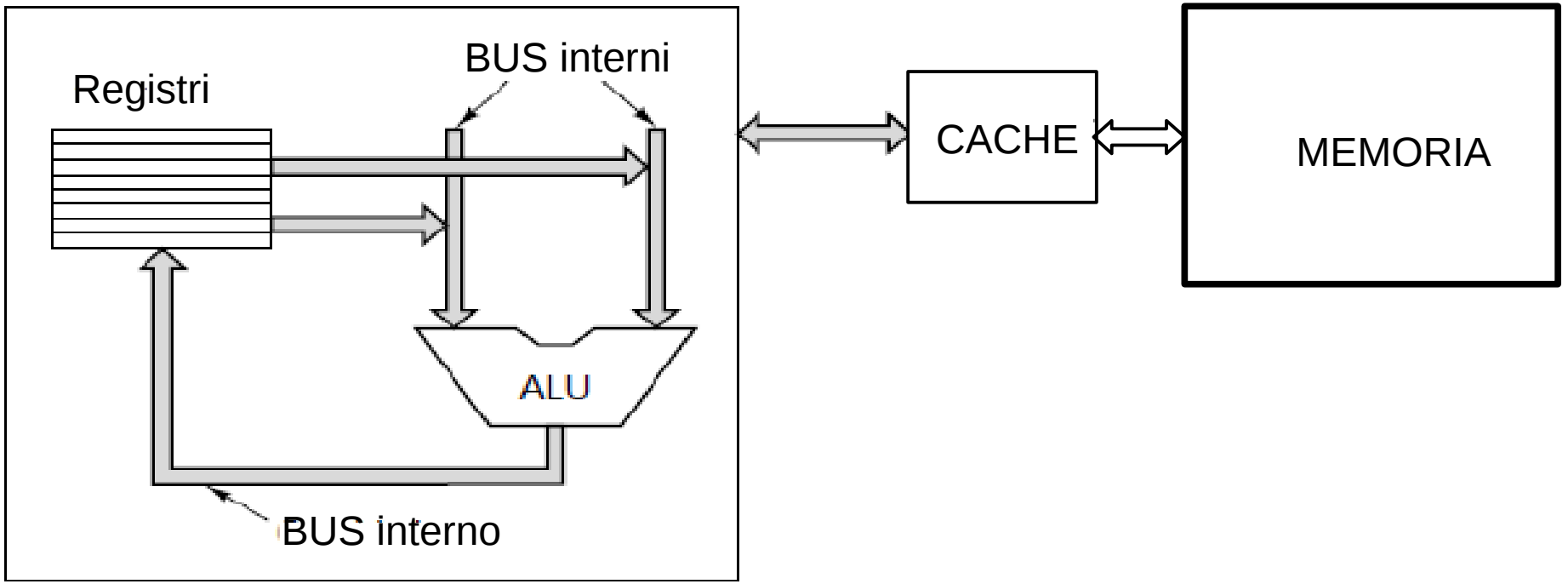
  Esempio: programma assembler:

```
        MOVE.W   0,R0
Loop:   MOVE.W   0,R0(A0)
        ADD      1,R0
        CMP      10,R0
        BLT      Loop
```

- Il modo di reagire dipende dalla istruzione o microistruzione interrotta

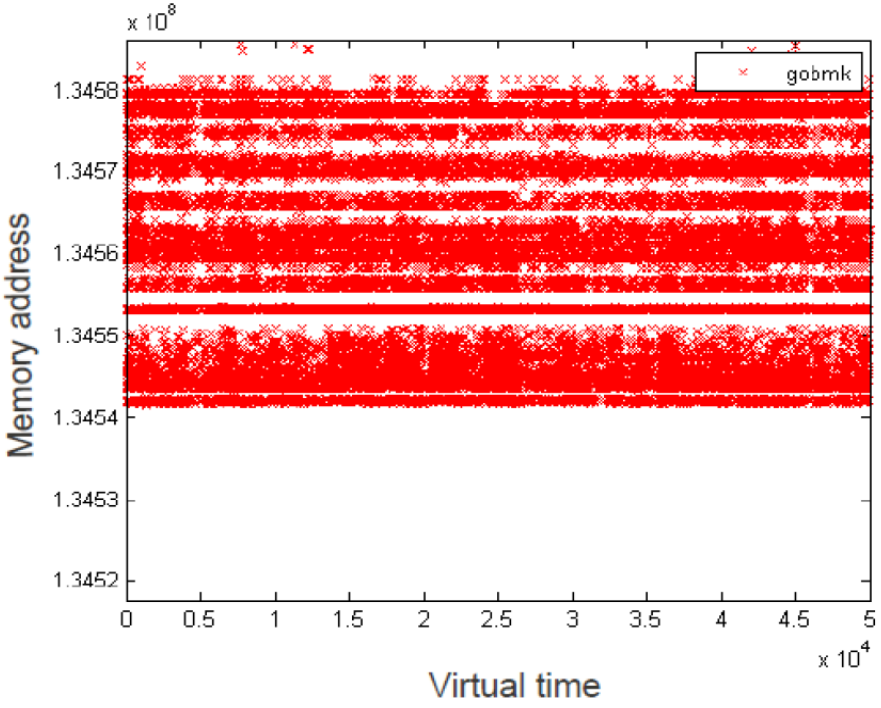
CPU

- Torniamo allo schema di massima
- La memoria (bus dati/indirizzi) viene acceduta dalla CPU che esegue un programma

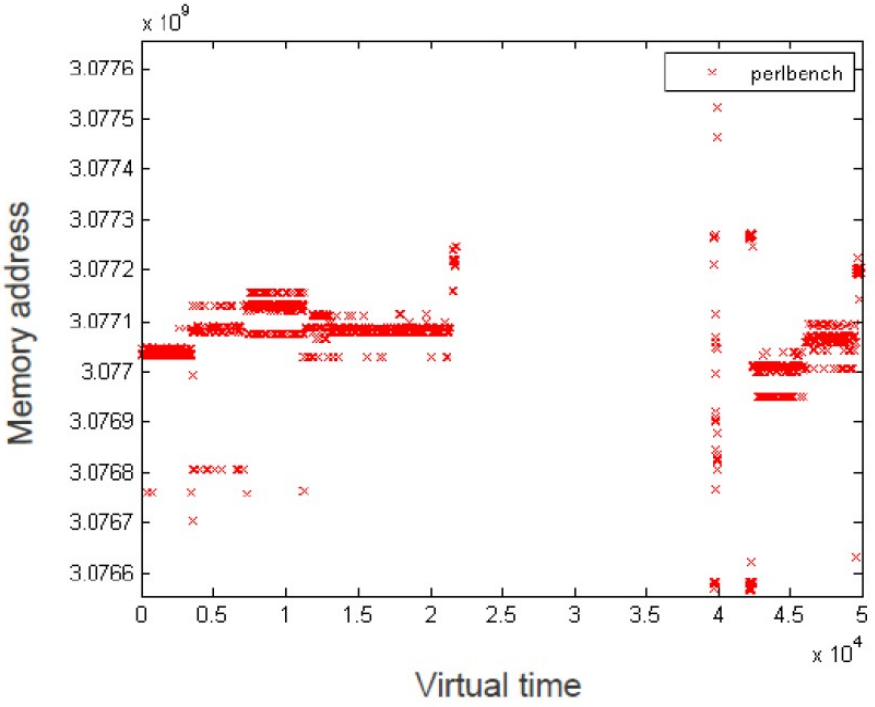


Accessi in memoria

- Andamento degli indirizzi nel tempo
- Dipende da tipo di esecuzione. Esempio:



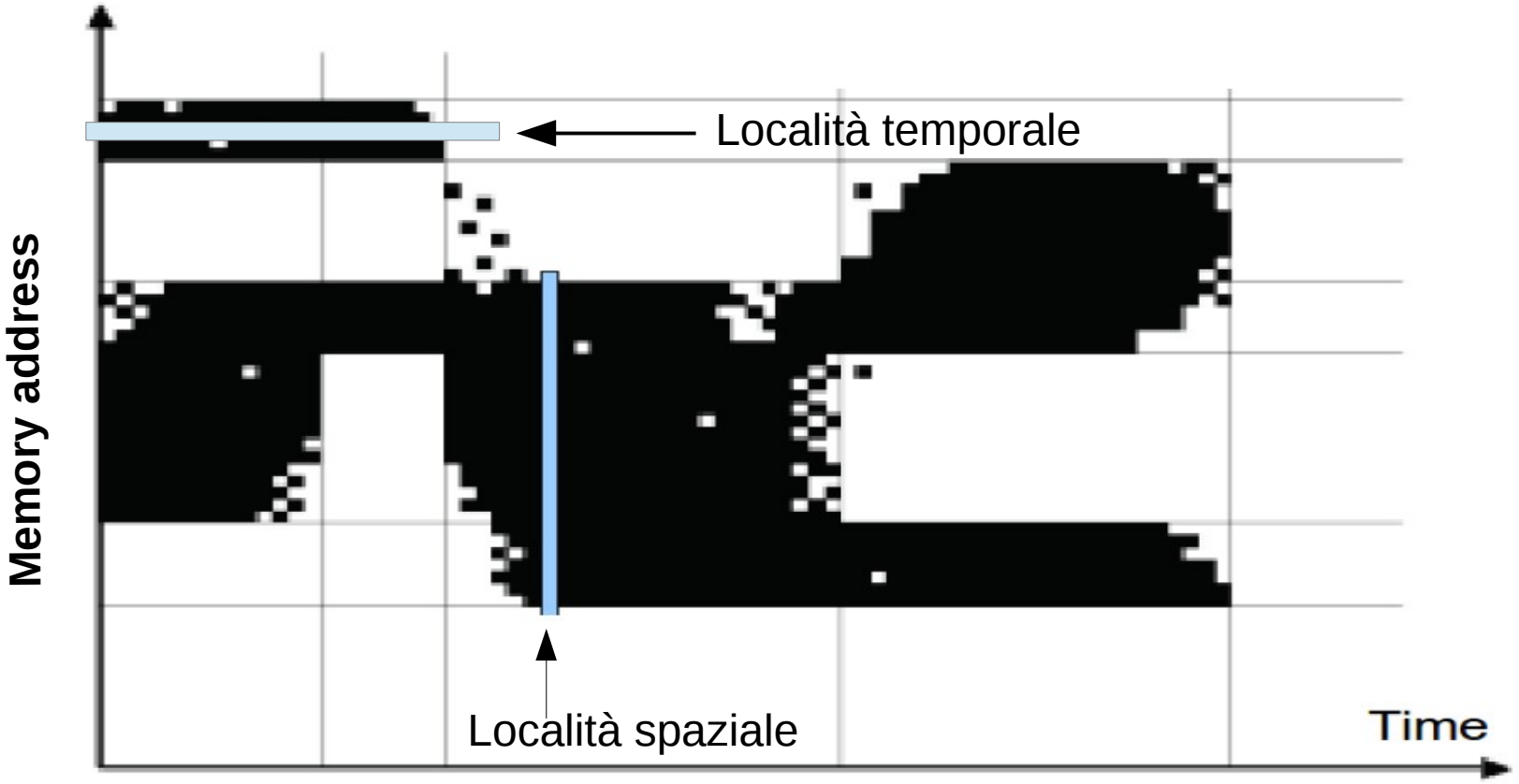
Gioco “GO”



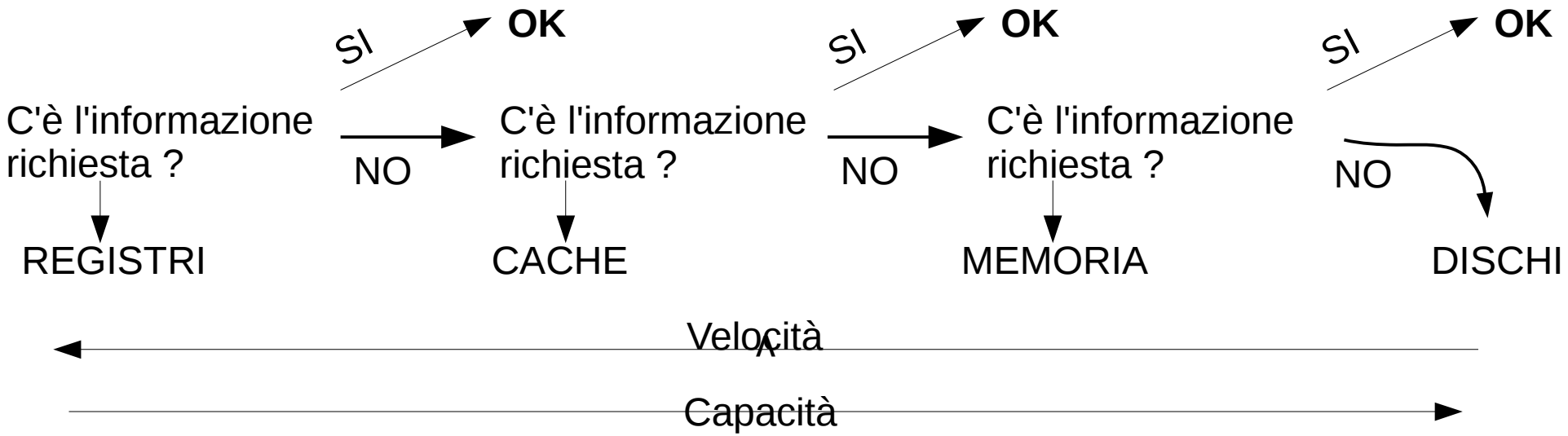
Compilatore Perl

Proprietà accessi: LOCALITA'

- Località spaziale: gli accessi tendono ad essere spazialmente raggruppati
- Località temporale: gli stessi indirizzi sono acceduti ripetutamente nel tempo



Gerarchie di memoria



- Lo scopo è di ridurre il tempo effettivo di accesso
- L'informazione è memorizzata nella memoria più lenta
- Viene quando richiesta caricata nella memoria più veloce
- Tutto questo funziona per via del Principio di Località degli accessi
- Hit/Miss

Gerarchie di memoria

- Hit rate = h = $\dfrac{Numero\ di\ successi}{Numero\ totale\ di\ richieste}$
- Miss rate = (1 - Hit rate)
- Tempo effettivo d'accesso =

  = $\dfrac{(Numero\ di\ Hit)\cdot Tempo\ di\ Hit + (numero\ di\ Miss)\cdot Tempo\ di\ Miss}{Numero\ totale\ di\ richieste}$

  = $h\cdot Tempo_{fast} + (1-h)\cdot(Tempo_{slow} + Tempo_{fast})$

CACHE L1-L2

- → M_1 → M_2 → $T_{eff} = h_1 T_1 + (1-h_1)(T_2+T_1) = h_1 T_1 + (1-h_1) T_2 + (1-h_1) T_1$

  → $\underline{T_{eff} = T_1 + (1-h_1) T_2}$

- Supponiamo di aggiungere in testa una cache M0 più veloce → M0 → M1 → M2

→ $T_{eff} = h_0 T_0 + (1-h_0)(T_0+T_1) = h_0 T_0 + (1-h_0)(T_0 + h_1 T_1 + (1-h_1)(T_1+T_2))$

$h_0 T_0 + (1-h_0) T_0 + (1-h_0) h_1 T_1 + (1-h_0)(1-h_1)(T_1+T_2)$

$T_0 + (1-h_0) h_1 T_1 + (1-h_0)(1-h_1) T_1 + (1-h_0)(1-h_1) T_2$

→ $\underline{T_{eff} = T_0 + (1-h_0) T_1 + (1-h_0)(1-h_1) T_2}$