# I processori

M. Sonza Reorda

Politecnico di Torino
Dip. di Automatica e Informatica

# Introduzione

**Un processore è un modulo che può corrispondere**

- **a una parte di un circuito integrato (*core*)**
- **a un circuito integrato a se stante.**

**Esistono numerose tipologie di processori in termini di complessità, potenza di calcolo, architettura, ecc.**

# Introduzione

**Un processore è un dispositivo che compie 2 tipi di operazioni:**

- **esegue istruzioni (il cui codice sta in memoria)**
- **interagisce con il mondo esterno (attraverso opportune interfacce).**

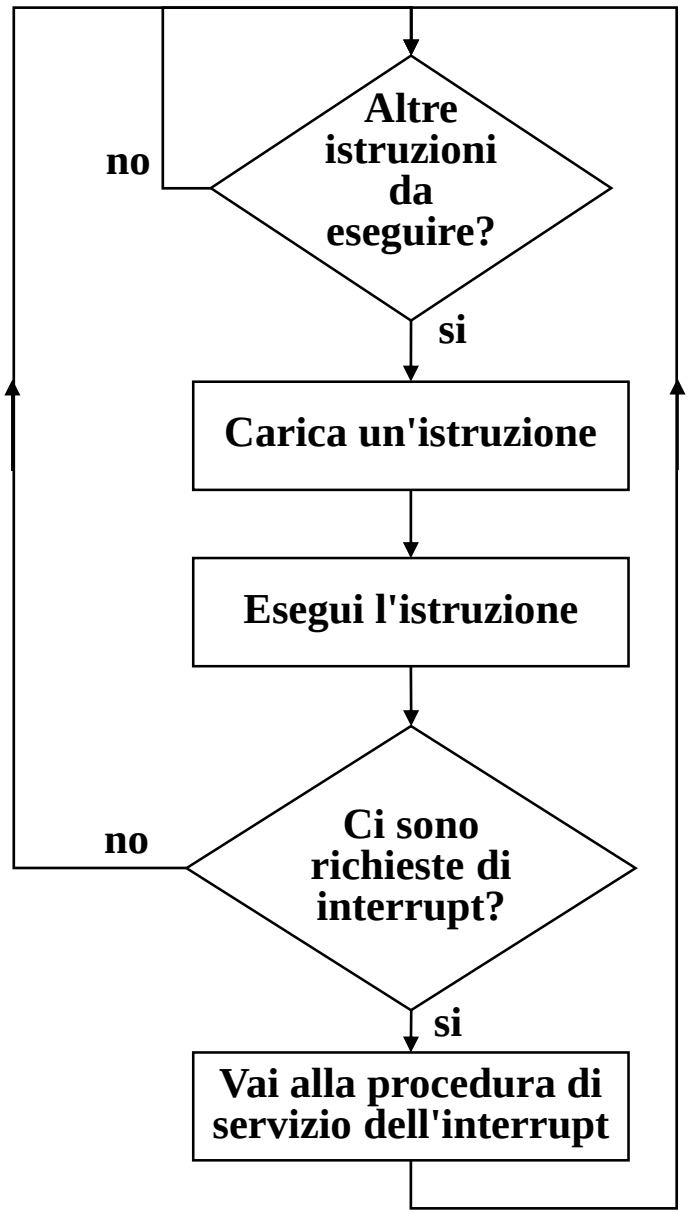
# Gestione di eventi esterni

**Il processore partecipa normalmente a tutte le operazioni che si svolgono sul bus del sistema:**

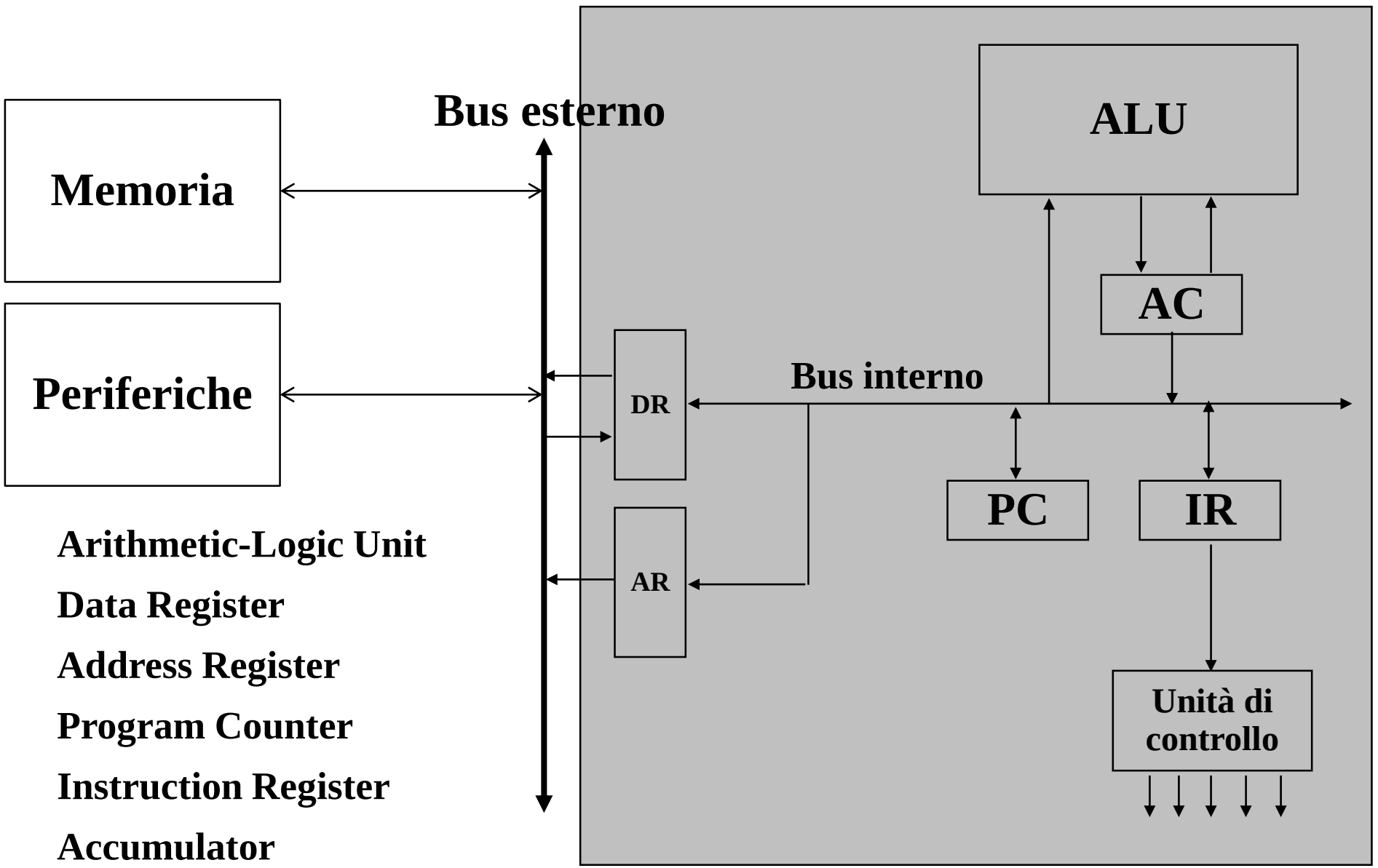
- **ad alcune partecipa direttamente (ad esempio facendo accesso alla memoria o ai dispositivi periferici)**
- **ad altre partecipa semplicemente prendendo parte al meccanismo di arbitraggio del bus.**

**Particolare importanza ha il meccanismo dell'*interrupt*, attraverso il quale un dispositivo esterno richiede l'esecuzione di una procedura di servizio.**

# Comportamento di un processore



# Architettura minima di una CPU




**ALU** Arithmetic-Logic Unit

**DR** Data Register

**AR** Address Register

**PC** Program Counter

**IR** Instruction Register

**AC** Accumulator

# Istruzioni

**L’esecuzione di ciascuna istruzione macchina si compone di 2 fasi:**

- ***fetch*: il codice dell’istruzione viene letto dalla memoria**
- ***execute*: il codice viene prima decodificato, poi eseguito. Questo comporta normalmente l’accesso ad uno o più operandi, l’esecuzione di una operazione su di essi, la scrittura del risultato.**

**La combinazione delle due fasi si dice *ciclo di istruzione*.**

# Accesso alla memoria

**Avviene in due fasi:**

1. **Il processore mette sul bus indirizzi l'indirizzo della cella di memoria e sul bus di controllo gli opportuni segnali di controllo**
2. **La memoria mette sul bus dati il contenuto della cella indirizzata (*lettura*), oppure**

   **Il processore mette sul bus dati il contenuto da scrivere nella cella indirizzata, che viene acquisito dalla memoria (*scrittura*).**

# Registri

**Il tempo per accedere alla memoria è normalmente superiore al tempo necessario alla CPU per processare i dati.**

**L'accesso alla memoria rappresenta quindi un *collo di bottiglia* per le prestazioni delle CPU.**

**Per questa ragione all'interno della CPU sono presenti alcune celle di memoria particolarmente veloci, note come *registri*.**

**Ove possibile le operazioni vengono svolte utilizzando i registri per contenere operandi e risultato.**

# Microistruzioni

**L’esecuzione delle 2 fasi di un'istruzione corrisponde all’attivazione di un certo numero di *microistruzioni* (corrispondenti ad esempio al trasferimento di un valore da un registro ad un altro).**

**Il tempo per l’esecuzione di una microistruzione è definito come *CPU cycle time* ($t_{CPU}$) e spesso coincide con il ciclo di clock del processore.**

**Nei processori tradizionali ciascuna istruzione richiede quindi un tempo pari ad un certo numero di colpi di clock.**

# Microistruzioni

**Fetch:**

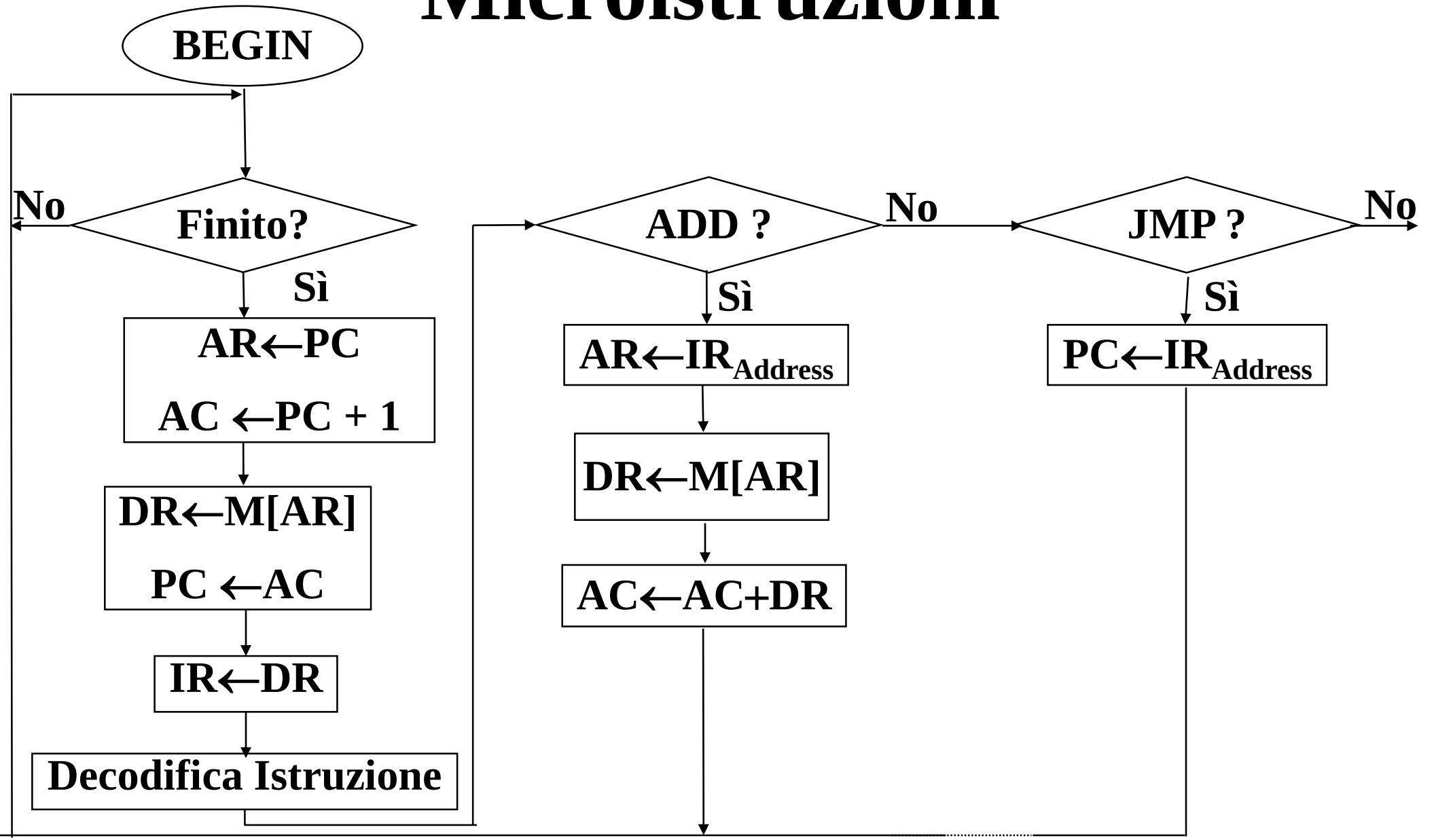
1. **AR←PC**
2. **DR←M[AR]**
3. **IR←DR**
4. **PC←PC+1**
5. **Decodifica Istruzione**

# Microistruzioni (cont.)

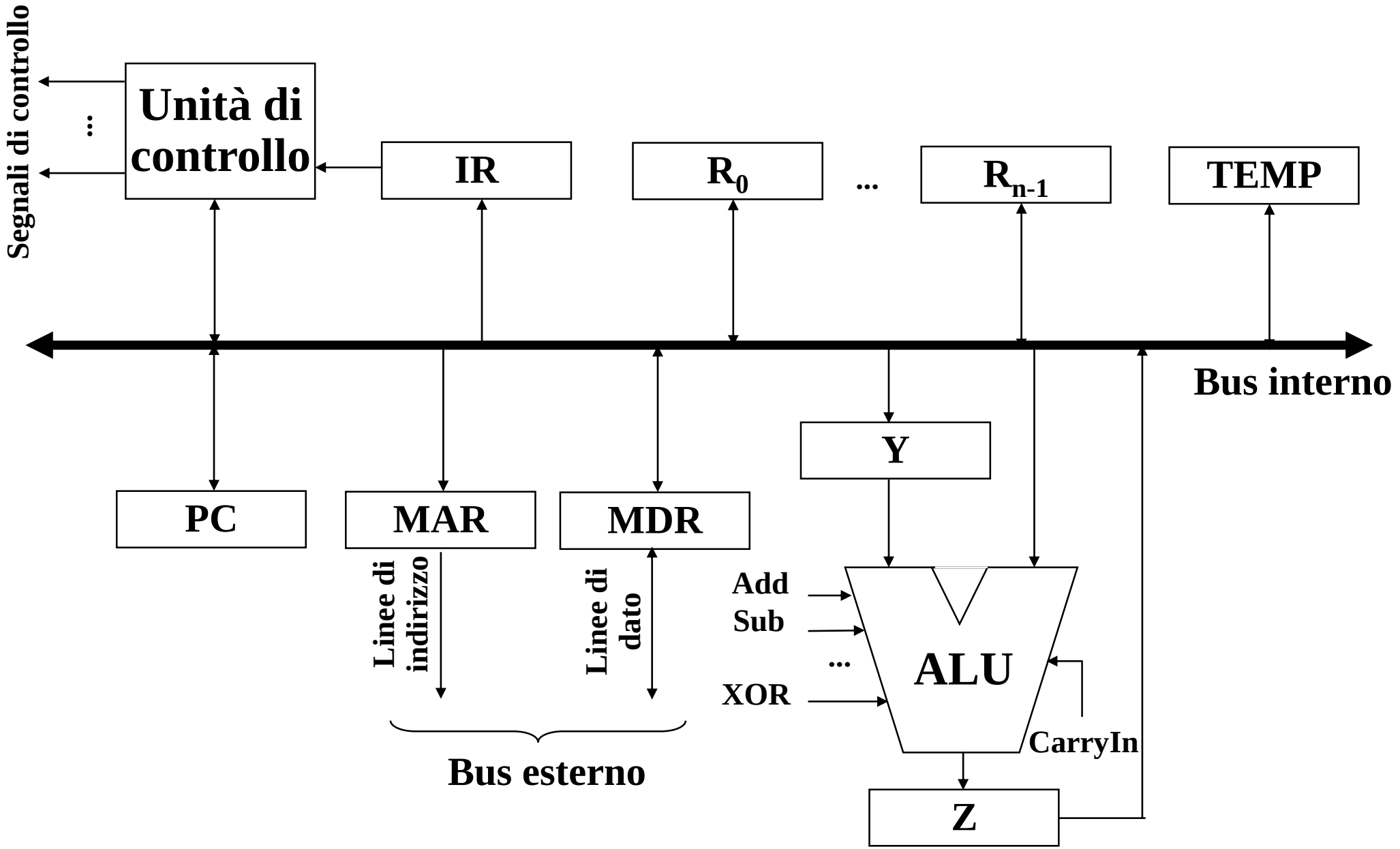
**Sfruttando il fatto che durante la lettura il bus locale non viene utilizzato è possibile ottimizzare le microistruzioni effettuate durante il fetch.**

1. **AR←PC, AC ←PC + 1**
2. **DR←M[AR], PC ← AC**
3. **IR←DR**
4. **Decodifica Istruzione**

# Microistruzioni



# Architettura di una CPU (II)



# Input/Output

**L'accesso ai dispositivi di I/O viene realizzato in maniera simile a quanto fatto per l'accesso alla memoria: ad ogni dispositivo di I/O è associato un *dispositivo di interfaccia*, che contiene un certo numero di registri (anche detti *porte*) visibili alla CPU attraverso il bus.**

**Ad ogni registro di interfaccia corrisponde un indirizzo, sul quale la CPU può eseguire opportune operazioni di lettura o scrittura.**

# Esempio

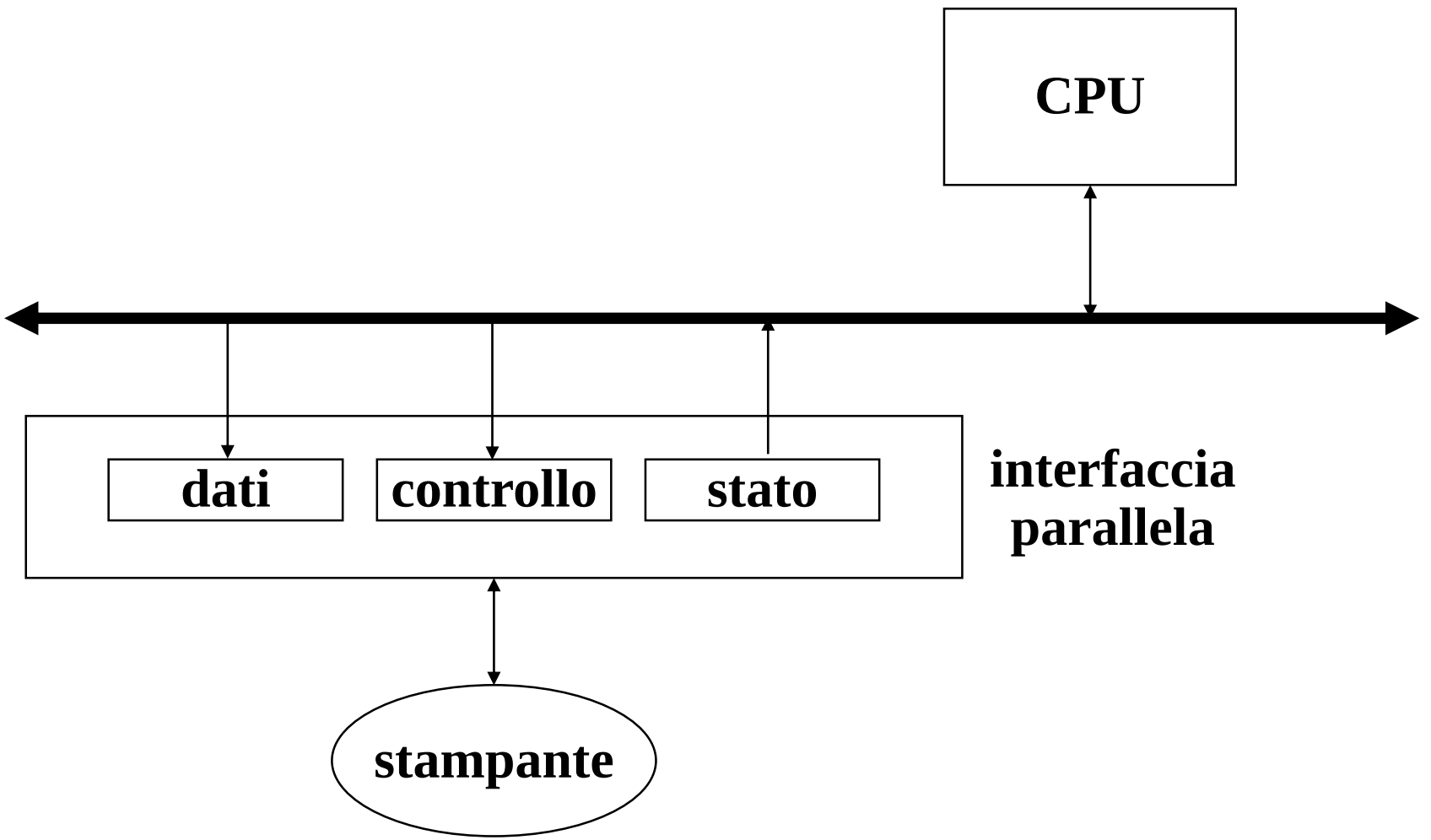
**Si consideri la connessione di una *stampante*, realizzata normalmente attraverso un'interfaccia parallela.**

**La stampante sarà vista dalla CPU come 3 registri corrispondenti ad altrettanti indirizzi:**

- **un registro per i *dati* (output)**
- **un registro per le informazioni di *controllo* (output)**
- **un registro per le informazioni di *stato* (input).**

**Per far eseguire delle operazioni alla stampante la CPU eseguirà delle scritture o letture sui 3 registri.**

# Esempio


CPU
dati
controllo
stato
interfaccia
parallela
stampante

# Estensioni

**Rispetto allo schema di base di un processore sono normalmente adottate alcune estensioni relative ai seguenti punti:**

- **registri**
- **unità aritmetiche**
- **status register**
- **stack.**

# Registri

**Il numero dei registri influenza fortemente le prestazioni di un processore. Quando sono molti si parla di *register-file* o *scratch-pad memory*.**

**È possibile che per ogni registro venga imposto un particolare uso; si possono avere**

- **registri dati**
- **registri indice**
- **registri contatore.**

# Registri

**Il numero dei registri infl**
**un processore. Quando s**
***scratch-pad memory*.**

**È possibile che per o registro venga imposto un particolare uso; si pos ono avere**

- **registri dati**
- **registri indice**
- **registri contatore.**

**Registri destinati a memorizzare dei dati**

# Registri

**Il numero dei registri influenza fortemente le prestazioni di un processore. Quando sono molti si parla di *register-file* o *scratch-pad memory*.**

**È possibile che per ogni particolare uso; si possono av**

- **registri dati**
- **registri indice**
- **registri contatore.**

**Registri destinati a memorizzare degli indirizzi in memoria, corrispondenti a celle in cui si trovano gli operandi**

# Registri

**Il numero dei registri influenza fortemente le prestazioni di un processore. Quando sono molti si parla di *register-file* o *scratch-pad memory*.**

**È possibile che per ogni registro venga imposto un particolare uso; si possono avere**

- **registri dati**
- **registri indice**
- **registri contatore.**

**Registri destinati ad eseguire operazioni di conteggio**

# Unità aritmetiche

**Possono essere più o meno potenti, a seconda delle operazioni che implementano:**

- **somma e sottrazione in fixed-point**
- **moltiplicazione e divisione in fixed-point**
- **operazioni in floating-point.**

# Registro di stato

**È un registro particolare contenente due gruppi di bit (o *flag*) denominati**

- **bit di stato**
- **bit di controllo.**

# Bit di stato

**I bit di stato sono**

- **forzati dal verificarsi di particolari condizioni (*carry*, *overflow*, risultato positivo o negativo, risultato nullo, parità) a seguito dell'esecuzione delle istruzioni**
- **testati dalle istruzioni di salto condizionato (e da eventuali altre).**

**In questo modo si implementano a livello di codice assembler i salti condizionati.**

# Esempio

**Codice C**

```
if(a == b)
      a = 1;
```

**Codice assembler 80x86**

```
        MOV    AX, a
        MOV    BX, b
        SUB    AX, BX
        JNZ    dopo
        MOV    AX, 1
dopo:   MOV    a, AX
        ...
```

# Esempio

**Codice C**

```
if(a ==
      a = 1;
```

**Codice assembler 80x86**

```
        MOV   AX, a
        MOV   BX, b
        SUB   AX, BX
        JNE   dopo
        MOV   AX, 1
dopo:   MOV   a, AX
        ...
```

**Forza a 0 o 1 il flag di zero, a seconda che il risultato sia nullo o meno**

**Salta se il flag di zero vale 0**

# Bit di controllo

**Controllano il comportamento del processore rispetto ad un determinato aspetto.**

**Ad esempio**

- **il *bit di interrupt* controlla la sensibilità agli interrupt**
- **il *bit di trap* fa funzionare o meno il processore in modalità debug.**

**Tali bit sono forzati a 0 o 1 da apposite istruzioni.**

# Stack

**È una struttura LIFO allocata in memoria.**

**Molti processori possiedono:**

- **un registro special-purpose, denominato *Stack Pointer* (SP) che punta all'elemento top**
- **due istruzioni (PUSH e POP) che manipolano implicitamente lo SP.**

# Stack: usi

**Lo stack viene usato:**

- **per le chiamate alle procedure (normali o di servizio dell'interrupt)**
- **per il passaggio di parametri alle procedure**
- **per la memorizzazione di variabili locali.**

# Soluzioni alternative

**In taluni casi (ad esempio nei sistemi *IBM 360-370*) il salvataggio dell'indirizzo di ritorno veniva effettuato utilizzando una particolare locazione di memoria.**

**In altri casi (ad esempio nell'architettura *PowerPC*) si usa un particolare registro.**