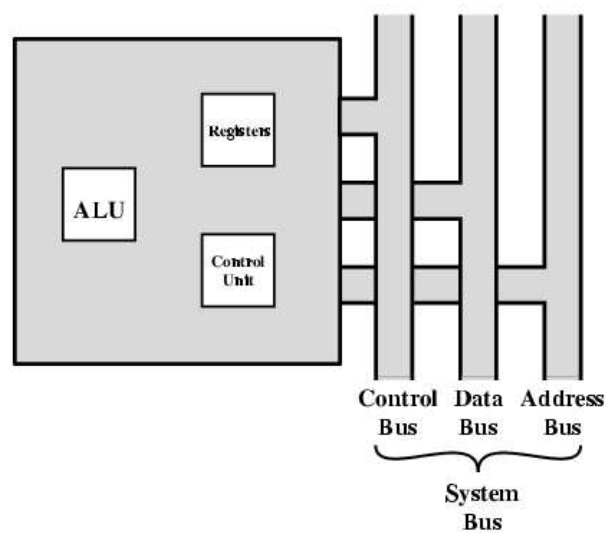


Struttura CPU

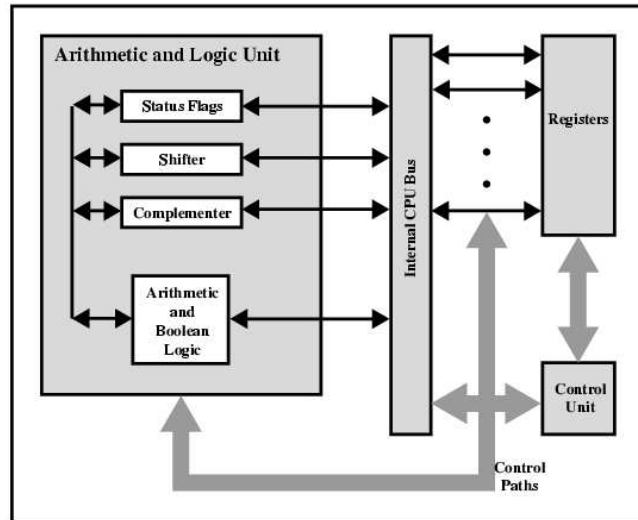
Compiti CPU:

- Prelevare istruzioni
- Interpretare istruzioni
- Prelevare dati
- Elaborare dati
- Scrivere (memorizzare) dati

CPU con bus di sistema



Struttura interna CPU



Architettura degli elaboratori -1

Pagina 13

Registri

- CPU ha bisogno di uno “spazio di lavoro” dove memorizzare i dati
- Questo “spazio di lavoro” è costituito dai **registri**
- Numero e funzioni svolte dai registri varia a seconda dell’impianto progettuale della CPU
- Scelta progettuale molto importante
- I registri costituiscono il livello più alto della così detta “Gerarchia della memoria”

Architettura degli elaboratori -1

Pagina 14

Registri

- Registri utente
 - usati dal “programmatore” per memorizzare internamente alla CPU i dati da elaborare
- Registri di controllo e di stato
 - usati dall’unità di controllo per monitorare le operazioni della CPU
 - usati dai programmi del Sistema Operativo (SO) per controllare l’esecuzione dei programmi

“programmatore”



1. Umano che programma in assembler
2. Compilatore che genera codice assembler a partire da un programma scritto in un linguaggio ad alto livello (C, C++, Java,...)

Ricordarsi che un programma in assembler è trasformato in codice macchina dall’assemblatore (+ linker) che trasforma il codice mnemonico delle istruzioni in codice macchina

Registri visibili all'utente: registri utente

- Ad uso generale
- Per la memorizzazione di dati
- Per la memorizzazione di indirizzi
- Per la memorizzazione di codici di condizione

Registri ad uso generale

- Possono essere veramente ad uso generale
- ...oppure dedicati a funzioni particolari
- Possono essere usati per contenere dati o indirizzi
- Dati
 - Ad esempio: accumulatore
- Indirizzi
 - Ad esempio: indirizzo base di un segmento di memoria



Segmento di memoria

- La memoria principale può essere organizzata, dal punto di vista logico (cioè concettuale), come un insieme di segmenti o spazi di indirizzamento multipli:
 - “visibili” al “programmatore”, che riferisce logicamente una locazione di memoria riferendo il segmento e la posizione della locazione all’interno del segmento:
es. segmento 4, locazione 1024
 - come supporto a questa “visione” della memoria, occorre poter indicare **dove**, all’interno della memoria fisica, inizia il segmento (*base*) e la sua **lunghezza** (*limite*)
es. il segmento 4 ha base = 00EF9445_{hex} e limite = 4MB
 - quindi occorrono dei registri dove memorizzare tali informazioni

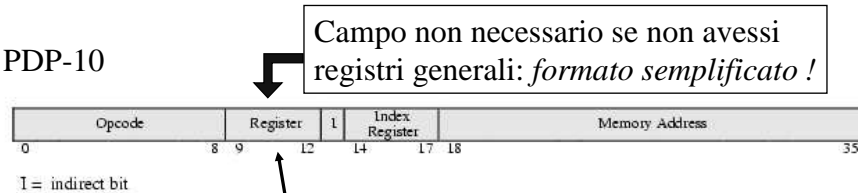
Registri ad uso generale

- Registri veramente ad uso generale
 - Aumentano la flessibilità e le opzioni disponibili al programmatore “a basso livello”
 - Aumentano la dimensione dell’istruzione e la sua complessità (perché ?)
- Registri specializzati
 - Istruzioni più piccole e più veloci
 - Meno flessibili

Perché aumenta dimensione e ↑ complessità ?

- Facciamo l'esempio di istruzioni a formato fisso:

PDP-10



Formato più lungo! 37 bit

4 bit → $2^4 = 16$ registri generali
Se si avessero 32 registri generali sarebbero necessari 5 bit di indirizzamento

Quanti registri generali?

- Tipicamente tra 8 e 32
- Meno di 8 = più riferimenti (accessi) alla memoria principale (perché?)
- Più di 32 non riducono i riferimenti alla memoria ed occupano molto spazio nella CPU
- Nelle architetture RISC tipicamente si hanno più di 32 registri generali

Perché più accessi ?



ESEMPIO: supponiamo di dover calcolare:

$\text{mem}[4] = \text{mem}[0] + \text{mem}[1] + \text{mem}[2] + \text{mem}[3]$

$\text{mem}[5] = \text{mem}[0] * \text{mem}[1] * \text{mem}[2] * \text{mem}[3]$

$\text{mem}[6] = \text{mem}[5] - \text{mem}[4]$

4 registri

```

R0 ← mem[0];
R1 ← mem[1];
R2 ← mem[2];
R3 ← mem[3];
R0 ← R0+R1;
R0 ← R0+R2;
R0 ← R0+R3;
R1 ← R1*R2;
R1 ← R1*R3;
R2 ← mem[0];
R1 ← R1*R2;
R0 ← R1-R0;
  
```

7 op
5 mem

6 registri

```

R0 ← mem[0];
R1 ← mem[1];
R2 ← mem[2];
R3 ← mem[3];
R4 ← R0+R1;
R4 ← R2+R4;
R4 ← R3+R4;
R5 ← R0*R1;
R5 ← R2*R5;
R5 ← R3*R5;
R0 ← R5-R4;
  
```

7 op
4 mem

Architettura degli elaboratori -1

Pagina 23

Quanto lunghi (in bit) ?

- Abbastanza grandi da contenere un indirizzo della memoria principale
- Abbastanza grandi da contenere una “full word”
- E’ spesso possibile combinare due registri dati in modo da ottenerne uno di dimensione doppia
 - Es.: programmazione in C
 - double int a;
 - long int a;

Architettura degli elaboratori -1

Pagina 24

Registri per la memorizzazione di Codici di Condizione

- Insiemi di bit individuali
 - es. Il risultato dell'ultima operazione era zero
- Possono essere letti (implicitamente) da programma
 - es. "Jump if zero" (salta se zero)
- Non possono (tipicamente) essere impostati da programma

Registri di Controllo e di Stato

- Program Counter (PC)
- Instruction Register (IR)
- Memory Address Register (MAR)
- Memory Buffer Register (MBR)

Program Status Word

- Un insieme di bit
- Include Codici di Condizione
 - Segno dell'ultimo risultato
 - Zero
 - Riporto
 - Uguale
 - Overflow
 - Abilitazione/disabilitazione Interrupt
 - Supervisore

Modo Supervisore

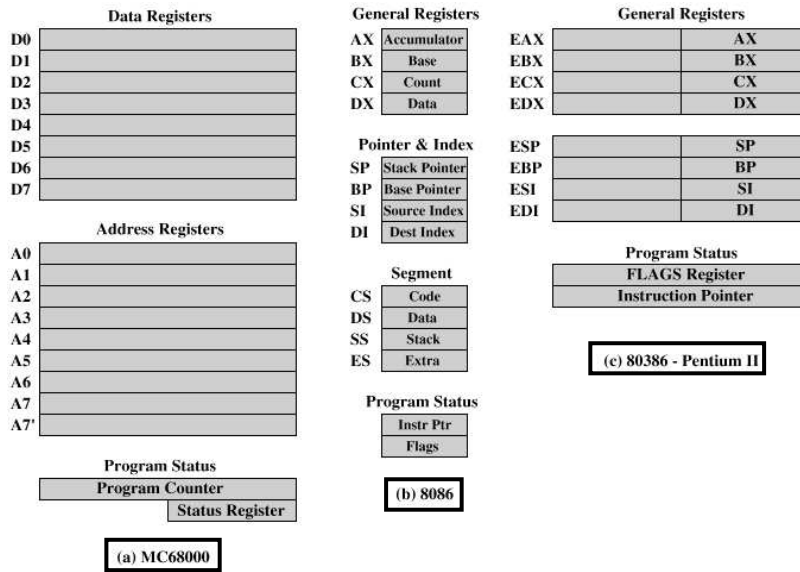
- Permette al sistema operativo di utilizzare le procedure del Kernel, che agiscono su componenti critiche del sistema
- In particolare permette l'esecuzione di istruzioni "privilegiate"
- Disponibile esclusivamente al sistema operativo
- Non disponibile all'utente programmatore
- Lo studierete in dettaglio nel corso di Sistemi Operativi

Altri registri

- Ci possono essere registri che puntano a:
 - Process control blocks (sistemi operativi)
 - Interrupt Vectors (sistemi operativi)
 - Tabella delle pagine della memoria virtuale

- La progettazione della CPU e quella dei sistemi operativi sono strettamente correlate

Esempi di Organizzazione di Registri



Ciclo esecutivo delle istruzioni: Fetch/Execute

- Lo avete visto nel corso di “Introduzione alle Architetture degli Elaboratori”
- Stallings, Capitolo 3
- Ne vediamo una versione revisionata

Indirettezza

- Per recuperare gli operandi di una istruzione può essere necessario accedere alla memoria
- La modalità di indirizzamento indiretto per specificare la locazione in memoria degli operandi richiede più accessi in memoria
- L'indirettezza si può considerare come un sottociclo del ciclo fetch/execute

