Architettura degli Elaboratori

Lezione 16 – Programmazione cablata, microprogrammazione, CISC e RISC

Giuseppe Cota
Dipartimento di Scienze Matematiche Fisiche e Informatiche
Università degli Studi di Parma

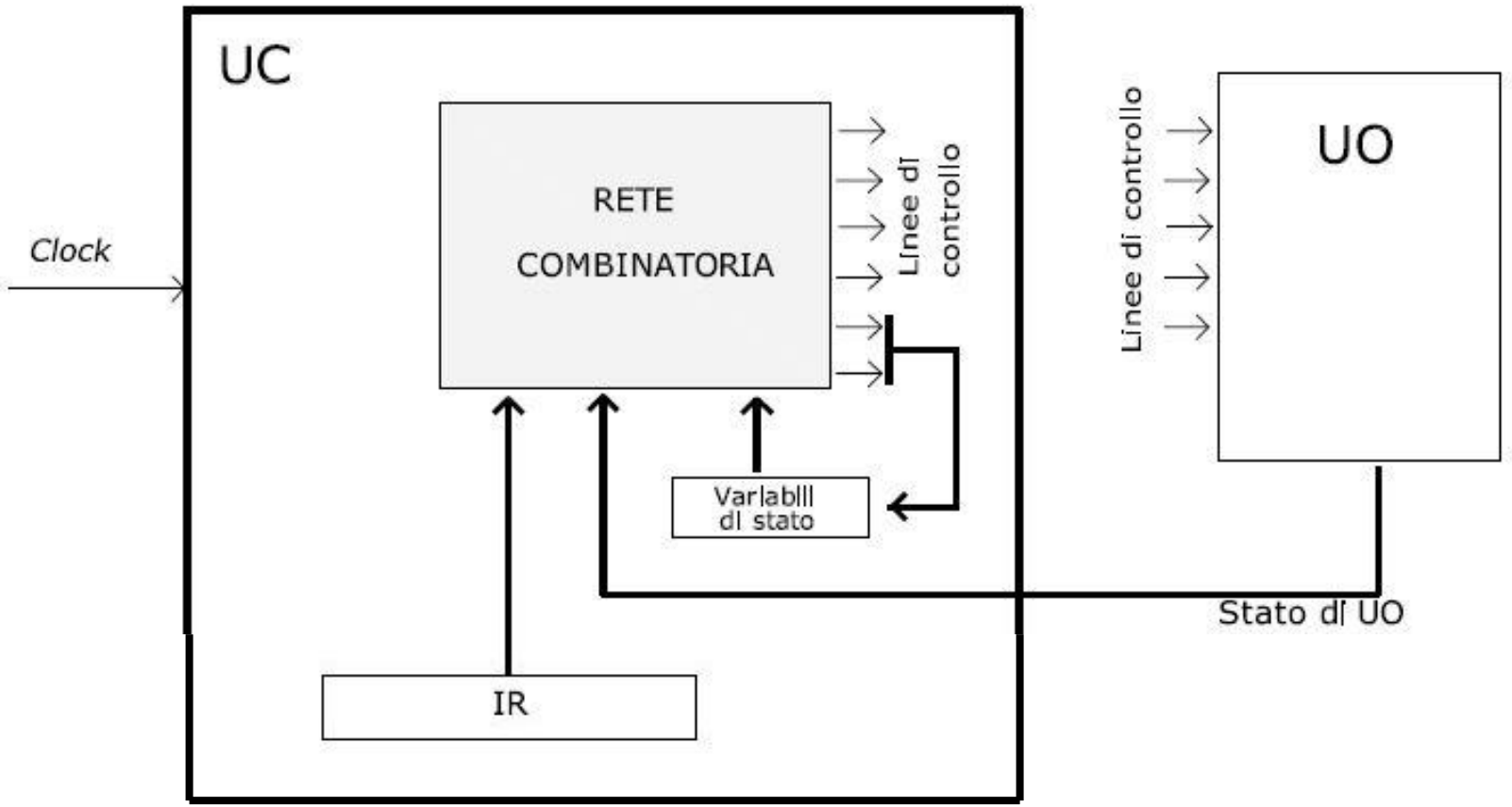
Indice

- Unità di controllo a logica cablata e microprogrammata
- RISC e CISC

Tipologie di unità di controllo

- La logica di controllo della CPU può essere realizzata in due forme:
  - **Cablata (hardwired)**
  - **Microprogrammazione**
- In realtà esistono anche approcci ibridi (moderni processori Intel).

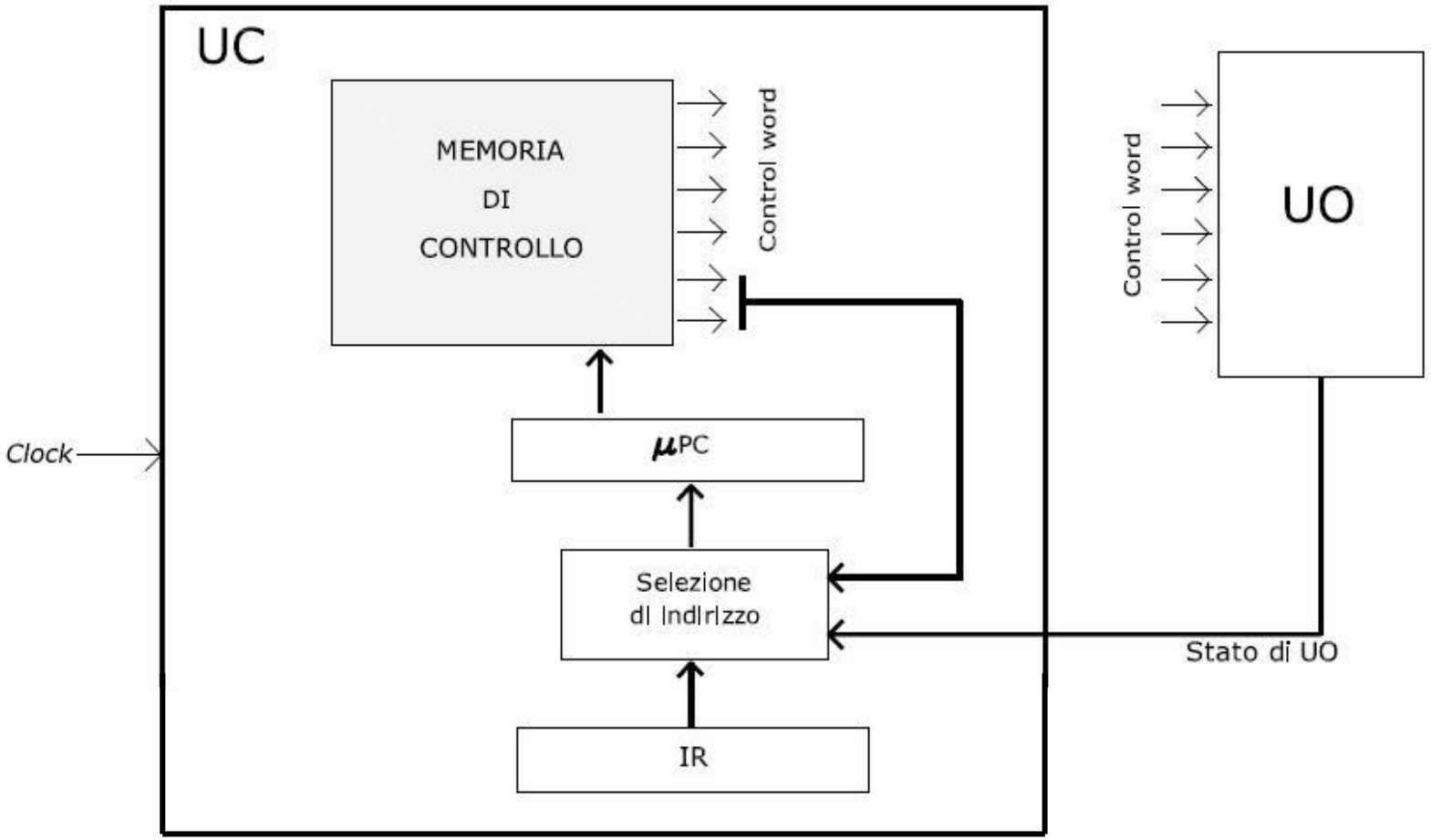
Unità di controllo a logica cablata



Unità di controllo a logica cablata

- Unità controllo in totale è rete sequenziale
- Un rete di decodifica combinatoria:
  - ingressi:
    - IR
    - Stato di UC
    - Stato di UO (e di eventuali altre parti esterne, ad es. memoria)
  - uscite: comandi temporizzati
    - Attiva vari comandi in diversi istanti temporali (sincronizzati dal clock)
- Progetto di una UC a logica cablata:
  - Uso di ROM. La rete combinatoria ha
    - Ingressi (indirizzi alla ROM): IR, stato di UO, stato di UC
    - Uscite: comandi,
  - ASIC (Application Specific Integrated Circuit)
  - FPGA (Field Programmable Gate Array)
  - SOC (System-On-a-Chip)
    - Composizione di diversi circuiti combinatori notevoli

Unità di controllo a logica microprogrammata



Unità di controllo a logica microprogrammata

- Tecnica affermatasi negli anni ‘70
- UC è una sorta di calcolatore nel calcolatore
- La memoria di controllo contiene le microistruzioni.
  - Realizzabile come ROM.
- μPC: contatore di microprogramma. Contiene l’indirizzo della prossima microistruzione
  1. All’inizio della fase di fetch μPC contiene l’indirizzo della memoria di controllo (I_0) del tratto di microprogramma corrispondente al fetch.
  2. Alla fine della fase di fetch μPC viene aggiornato con il contenuto (o una opportuna decodifica) di IR in modo da puntare alla microroutine (contenuta nella memoria di controllo) che effettua le azioni richieste dalla particolare istruzione.
  3. Durante l’esecuzione della microroutine è possibile che il μPC venga aggiornato in base alle microistruzioni.
  4. Al termine, μPC viene di nuovo caricato con (I_0)

Unità di controllo a logica microprogrammata

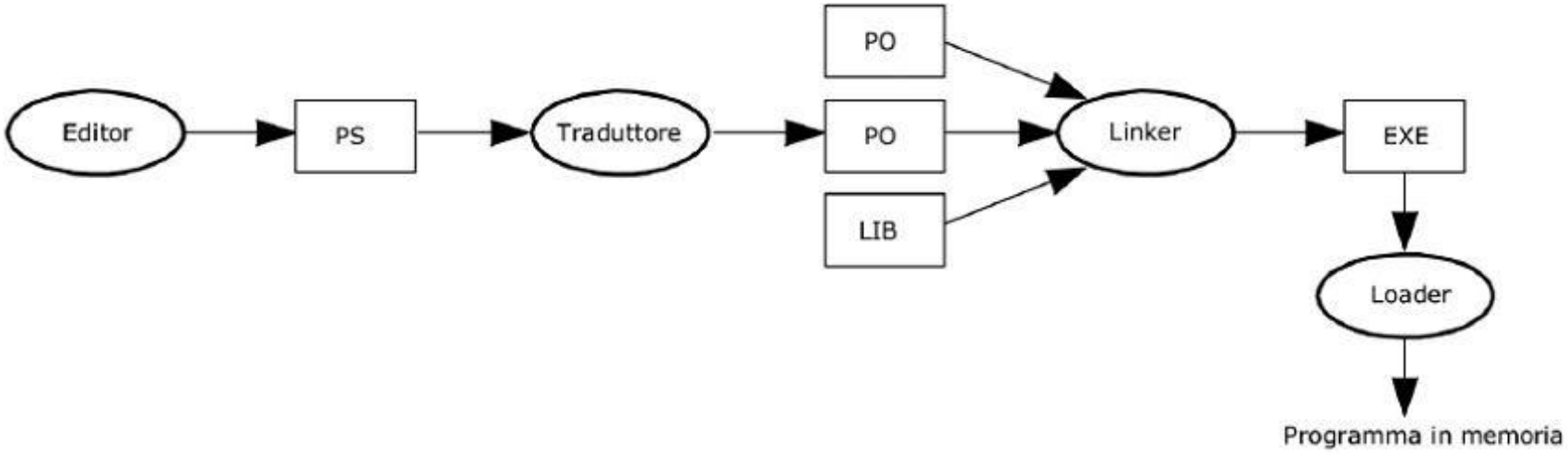
- Differenti soluzioni:
  - sequenza di *parole di controllo*, *micro control word* (*CW*) gestite da μPC
  - microistruzioni che contengono codificato al loro interno l'indirizzo della prossima microistruzione (senza μPC)
  - nanoprogrammazione: le microistruzioni sono a loro volta interpretate da una unità nanoprogrammata (68000)
- **Microprogrammazione orizzontale**:
  - ogni bit di una CW corrisponde ad una linea di comando
  - Cattivo impiego della memoria di controllo, avrò parole di controllo con tanti 0 e alcuni 1.
- **Microprogrammazione verticale**
  - le CW contengono i comandi in forma convenientemente codificata.
  - Le linee di controllo sono raggruppate in modo che linee appartenenti a uno stesso gruppo non possono essere asserite contemporaneamente.
    - Un gruppo è rappresentato dal numero di bit che servono per codificare le linee di appartenenza.
    - **Uso dei decoder**. Se un gruppo è composto da N linee di controllo, userò $\lceil \log_2 N \rceil$

Logica cablata e microprogrammazione
Evoluzione storica

- Fino a fine anni ‘60: logica cablata (PDP8, HP 2116)
- Negli anni ‘70 si diffuse la microprogrammazione
  - VAX 11/789 (DEC) S370/168 (IBM)
    - Oltre 400.000 bit di memoria di controllo
  - Microprocessori (8080, Z80, 8086, 6800)
    - Il 6800 aveva la nanoprogrammazione, ovvero le istruzioni del microprogramma erano a loro volta interpretate da un nanoprogramma.
  - **Repertorio di istruzioni molto esteso e variato: CISC**
- Dagli anni ‘80 si è riaffermata la logica cablata
  - A volte un misto delle due (processori Intel)
  - **Affermazione delle macchine RISC**
  - Evoluzione dell’architettura Intel: da CISC a (praticamente/quasi) RISC

CISC e RISC

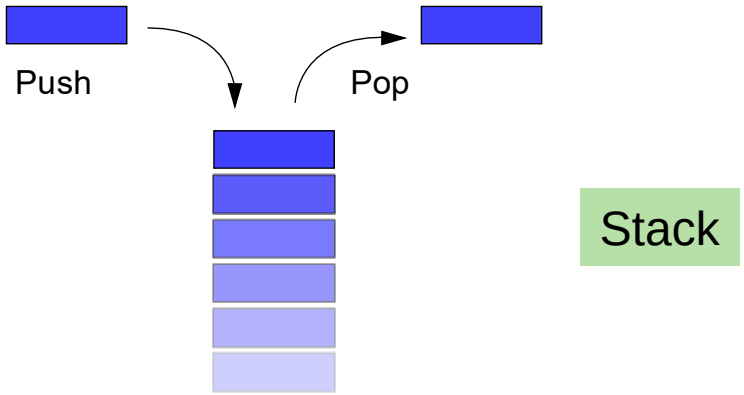
Programmi e repertorio delle istruzioni



PS programma sorgente
PO programma oggetto
LIB sottoprogrammi di libreria
EXE modulo in forma eseguibile

Sequenzializzazione delle istruzioni

- Le istruzioni vengono eseguite in sequenza, incrementando il PC (Program Counter) di quanto occupa l'istruzione eseguita
- Alcune istruzioni permettono il trasferimento del controllo (rompere la sequenzialità):
  - Istruzioni di salto condizionato o incondizionato
  - Istruzioni di chiamata (*call*). Prevedono il salvataggio dell'indirizzo di ritorno
    - *uso dello stack (pila):* pezzo di memoria centrale con logica LIFO (Last In First Out): i dati vengono estratti/letti (*pop*) in ordine rigorosamente inverso rispetto a quello in cui sono stati inseriti/scritti (*push*).
  - Inoltre, il normale flusso sequenziale può essere modificato dalle *interruzioni* (*uso dello stack*).

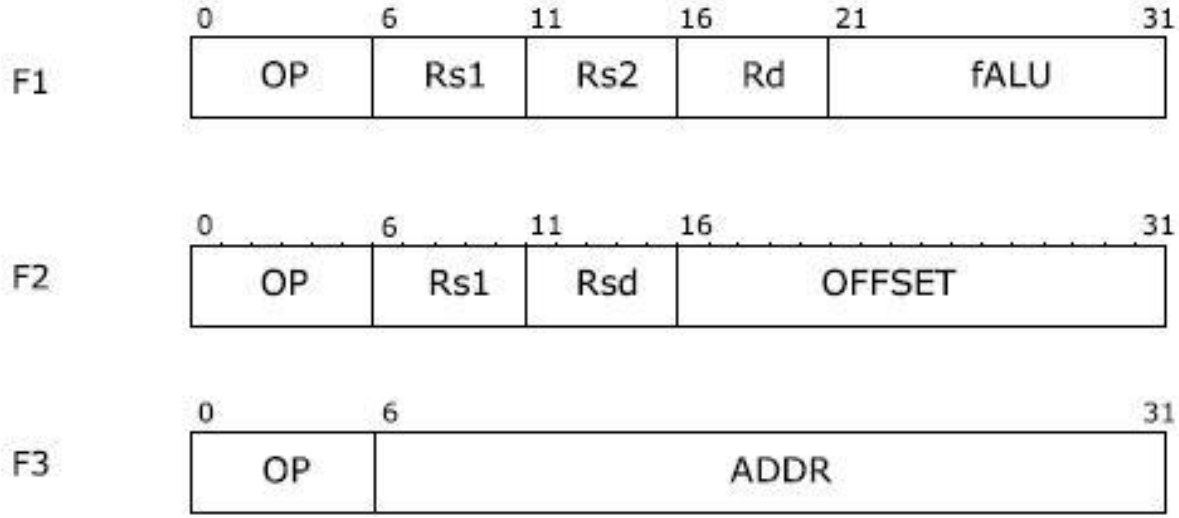


Repertorio delle istruzioni

- Il repertorio delle istruzioni macchina caratterizza un'architettura.
- Il numero di istruzioni nel repertorio, la potenzialità delle singole istruzioni, il loro formato e la loro codifica hanno grande influenza sulle prestazioni.
- Esistono due approcci per la definizione del repertorio delle istruzioni:
  - **Architetture CISC: Complex Instruction Set Computers**
  - **Architetture RISC: Reduced Instruction Set Computers**

Repertorio delle istruzioni stile RISC

- **RISC: Reduced Instruction Set Computers**
- Le istruzioni hanno tutte la stessa dimensione.
- Il campo del codice di operazione occupa un campo predefinito.
- Numero estremamente limitato di formati.
- Dato un formato, la posizione dei campi è sempre la stessa.



Aritmetiche/logiche

Load/Store

Salti condizionati

Salti incondizionati

Repertorio delle istruzioni stile CISC

- **CISC: Complex Instruction Set Computers**
- Le istruzioni non hanno dimensione fissa
- Il campo del codice di operazione occupa un numero variabile di bit
- Esiste un numero estremamente ampio di formati
- L'interpretazione di alcuni campi è condizionata da altri campi

Ragioni per le CISC

- Un repertorio di istruzioni esteso è preferibile perché:
  - Istruzioni potenti semplificano la programmazione
  - Riduce il gap tra linguaggio di macchina e linguaggio di alto livello
- L'uso efficiente della memoria (decenni fa era costosa (basata su nuclei elettromagnetici)) era la preoccupazione principale:
  - meglio avere codici compatti
- Essendo (tempo fa) la memoria di controllo molto più veloce della memoria centrale, portare funzionalità nella prima avrebbe migliorato le prestazioni della macchina.

(…Tuttavia) Ragioni contro le CISC

- Memorie RAM: molto più veloci delle precedenti a nuclei elettromagnetici
- Cache: riducono ulteriormente i tempi di esecuzione.
- Comportamento dei programmi:
  - l'80% delle istruzioni eseguite corrispondeva al solo 20% del repertorio.
  - Conviene investire nella riduzione dei tempi di esecuzione di quel 20%, anziché aggiungere raffinate istruzioni, quasi mai usate, ma responsabili dell'allungamento del tempo di ciclo di macchina.
  - Conviene costruire processori molto veloci, necessariamente con repertori semplici, e contare sull’ottimizzazione del compilatore.

CISC vs RISC

- Agli albori: logica cablata
- Anni ‘70: microprogrammazione
  - Repertori di istruzioni complessi, **CISC**
  - (si sperava di superare il *semantic gap* tra i linguaggi di programmazione di alto livello e il linguaggio macchina)
- Dagli anni ’80 si è riaffermata la logica cablata
  - Repertori di istruzioni semplici (ridotti), **RISC**
  - Ridurre al minimo la comunicazione con la memoria (che è lenta) => elevato numero di registri di CPU
  - Le istruzioni devo essere semplici: un’istruzione che fa aumentare del 10% il periodo di clock, deve produrre una riduzione di almeno il 10% dei cicli eseguiti
  - Si affida all’efficienza dei compilatori (ottimizzazione dei calcolatori).

CISC vs RISC

- Con le architetture RISC possiamo usare accorgimenti per velocizzare l'esecuzione (pipeline più efficienti).
- Con l'uso di memorie veloci e cache le istruzioni di macchina possono essere eseguite quasi alla stessa velocità delle microistruzioni, non c'è vantaggio nello spostare funzionalità a livello di microcodice.
  - È più facile modificare una libreria che una memoria di controllo.

Domande?

Riferimenti principali

- Capitoli 2 e 3 di ***Calcolatori elettronici. Architettura e Organizzazione,*** Giacomo Bucci. McGraw-Hill Education, 2017.