Architettura degli Elaboratori

Informatica per la Comunicazione Digitale

Università degli Studi di Milano

Lezione 12:

CPU e linguaggio MIPS
(parte A)

Marco Tarini

1

Architettura di Von Neumann (origine: anni '40)

Programma = sequenza di **istruzioni**.

Concetto base:

- Come abbiamo visto,
  le istruzioni sono codificate (in linguagio macchina) da parole di bit
- Istruzioni = solo un altro tipo di dato
- Memorizzare anche il programma nella RAM
- Cioè, il programma (una sequenza di istruzioni), che processa i dati, è anch'esso memorizzato della RAM
  - la stessa RAM dove tieniamo anche i dati da processare!
- **Software** = programma + dati (entrambi in RAM)
- Per eseguire il programma, l'eaboratore:
  - Legge dalla RAM un'istruzione (fase "fetch")
  - Interpreta e esegue l'istruzione (fase "execute")
  - Ripete da capo, per l'istuzione successiva

ciclo **fetch and execute**

2

Data Path e Control Unit

- L'*esecuzione* delle istruzioni è realizzata in un grande circuito detto «**Datapath**», una sorta di "percorso a gincana" per bits dove i dati…
  - "partono" da certe unità di memoria (dove erano stati memorizzati)
  - "viaggiano" su certi canali,
  - vengono elaborati dai vari blocchi funzionali che "attraversano",
  - e "arrivano" in certi altre unità di memoria, dove vengono finalmente memorizzati (fase datta «write back», che avviene alla fine del ciclo di clock)
- L'*interpretazione* delle istruzioni è realizzata da una «**Control Unit**»
  - Un circuito che fa da "cabina di regia", controllando attraverso appositi segnali («di controllo») il funzionamento del Datapath, in funzione dell'istruzione corrente

- 3 -

3

Schema per un archiettura di Von Neumann: I componenti di una CPU

- Il «**Program Counter**» (**PC**) è un **registro** speciale dedicato a mantenere l'indrizzo in RAM dell'istruzione corrente
- Un «**DataPath**» (come quello visto in embrione) provvisto di una (o più) **ALU**, il **banco di registri**, l'interfaccia verso la **RAM**, e varia logica di controllo
- Una «**Control Unit**», è solo un **circuito combinatorio** che, dato in input l'istruzione corrente (in linguaggio macchina), produce i **segnali di comando** da diramare al **DataPath**

- 4 -

4

Ciclo Fetch and Execute

1. Leggi l'istruzione corrente
   Lettura dalla RAM alla locazione presente in **Program Counter**
   Questa lettura restituisce l'**Istruzione Corrente**
2. Interpreta l'istruzione corrente
   La **Control Unit** elabora l'istruzione corrente e produce i segnali di comando al **Data path**
3. Esegui l'istruzione corrente da parte del «**Data path**», includendo:
   (a) Operazioni matematiche fra registri (utilizzando la **ALU**)
   (b) Richiesta di scrittura/lettura su **RAM** (da/a registri)
   (c) Calcolo del prossimo **Program Counter** (normalmente, incrementando il suo valore corrente)
4. Write-back (al termine del ciclo di clock!)
   La memoria e/o i registri memorizzano i valori prodotti nel passo 3.
   Anche il Program Counter assume il nuovo valore!
5. Riparti da 1

- 5 -

5

L'instruction set «MIPS»

- Da questo punto in poi, e fino alla fine del corso, useremo, per tutti i nostri esempi, un'ISA esistente: il MIPS
  - Che era stata introdotta nella prima lezione
- Vedremo:
  - Come codificare le istruzioni in **Linguaggio macchina MIPS**
  - La basi del funzionamento di una **CPU** (*) capace di eseguire programmi in **Linguaggio macchina MIPS**
  - La traduzione delle istruzioni in **Linguaggio assembler MIPS** (che, normalmente, sarebbe compito di un Assembler)
  - I tipi di dato previsti in un **Programma MIPS**
  - Qualche esperimento di programmazione in **Linguaggio assembler MIPS** (che, normalmente, sarebbe compito di un Compilatore da un linguaggio ad alto livello, come il Go)

(*) Per semplicità, vedremo solo una CPU detta «**a ciclo singolo**», cioè progettata per eseguire un'intera istruzione in un solo ciclo di clock

- 6 -

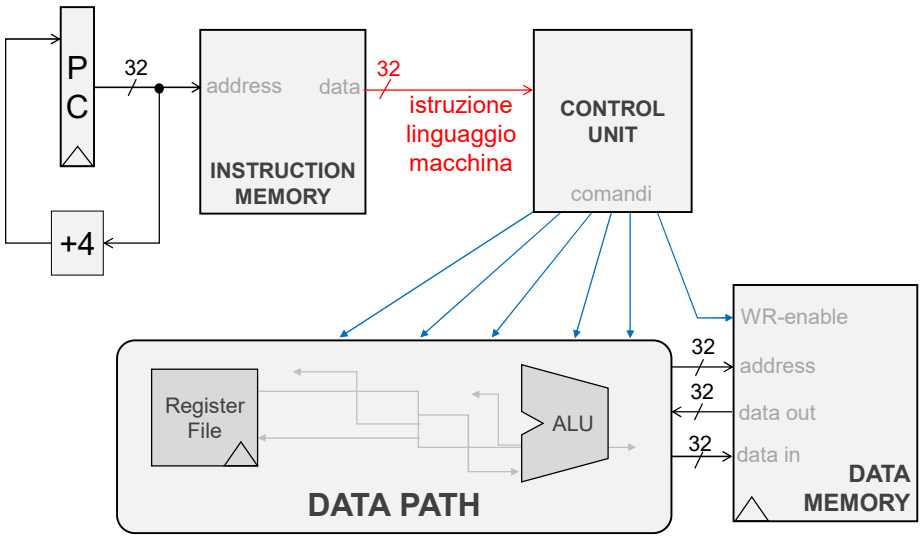
6

Alcune scelte base dell'Istruction Set MIPS

- Tutte le istruzioni sono codificate in **32** bit
- I registri sono costituiti da **32** bit
  - Cioè, le *parole* (word) sono di **32** bits
- Si hanno a disposizione **32** registri
  - Il registro R0 è speciale, e vale sempre 0
- Si usano operazioni con (al più) due operandi (di **32** bits)
  - I risultati delle operazioni sono parole di **32** bits
  - (eccezione: pochissime, come la moltiplicazione… producono *2* parole da **32** bits)
- Gli indirizzi di memoria sono di **32** bits
  - E in memoria si leggono o scrivono parole da **32** bits
  - Le parole sono indicizzate a livello di byte: ogni accesso quindi legge 4 bytes (4x8 = **32**): quello dell'indice fornito e i tre successivi

7

Ciclo Fetch and Execute in una CPU esempio (con scelte del MIPS)



8

Esercizi di riepilogo

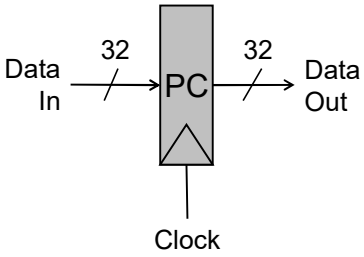
Nessun esercizio in questa parte, ma accertarsi
di aver ben chiari in mente i blocchi funzionali visti fin'ora,
che verranno utilizzati nel data path che stiamo per vedere, incluso:

- Register bank
- Memory bank
- ALU
- Ma anche blocchi più a basso livello:
  - MUX
  - Extender
  - DEC / COD
  - Addizionatore (con riporto di ingresso)
  - Registri paralleli

Nei prossimi lucidi, passiamo in rassegna
i componenti che stiamo per usare nella nostra CPU MIPS

9

Blocchi che stiamo per utilizzare: il Program Counter (registro parellelo)

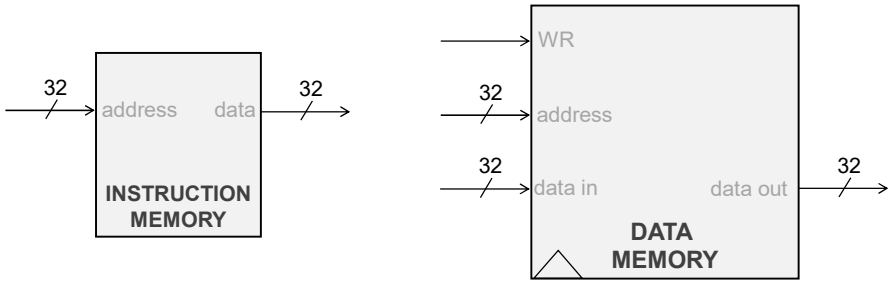


- **Program Counter**: un registro speciale in più, separato da quelli nel registre bank
- E' preposto ad un unico scopo: memorizzare l'indice dell'istruzione corrente (in memoria istruzioni).

10

Blocchi che stiamo per utilizzare: Due memorie RAM con indirizzamento a 32 bit

- Useremo due memorie centrali **separate**
  - Una per le **istruzioni** (che contiene il programma)
  - Una per i **dati**
- Necessario nel nostro design «a ciclo singolo», perché vogliamo poter eseguire due letture nello stesso ciclo di clock: il fetch dell'istruzione e l'eventuale operaz. RAM
  - Questa scelta non è adottata nel monde reale; è solo per semplicità
- Nota: nalla memoria di istruzioni non serva mai scrivere, ma solo leggere. Quindi ha solo il bus di uscita e non è sincronizzata

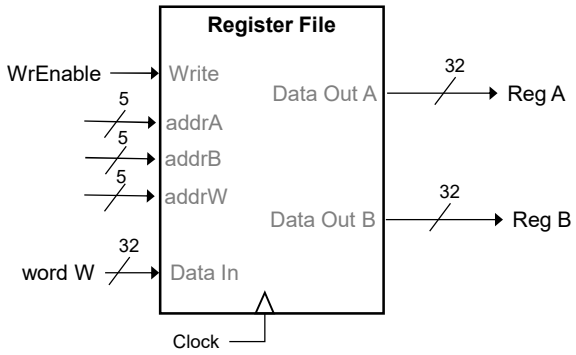


- 11 -

11

Blocco che useremo (ripasso) un Register File 32x32

- **Register File**:
  32 registri da 32 bit
  - Due bus di uscita
    - Reg A e Reg B
  - Un bus di input
    - Word W



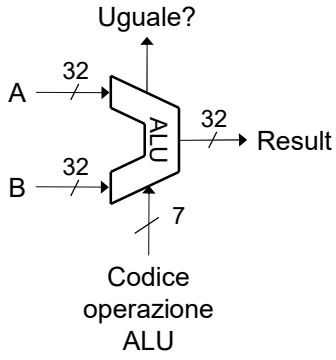
- Ingressi:
  - addrA e addrB selezionano i registri da riversare su RegA e RegB
  - addrW seleziona il registro in cui verranno scritti i dati provenienti da wordW quando Write è 1
  - Clock: significativo solo in scrittura. In lettura il circuito si comporta in modo combinatorio: su BusA e BusB ci sono sempre i valori contenuti nei registri indicati da RA e RB

- 12 -

12

Blocco che useremo (ripasso)
ALU

- Esegue opeazioni come: somma, sottrazione, or bitwise
- Produce esiti come «uguale?» nelle operazioni di confronto
- Codice a 7 bit, massimo 2^7 = 128 operazioni diverse
  di solito sono meno

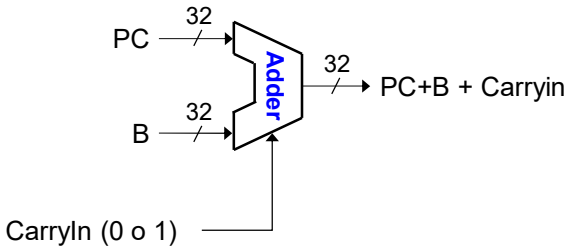


- 13 -

13

Blocco che useremo (ripasso)
Addizionatore

- Sarà usato per incremento del PC
- E' necessario un circuito separato dalla ALU!
  - Per lo stesso motivo per cui abbiamo due memorie separate: durante un ciclo di clock dobbiamo poter eseguire due somme: incremento del PC di 4, e eventuale operazione matematica



- 14 -

14

Sign-extension Vs zero extension (ripasso)

In binario

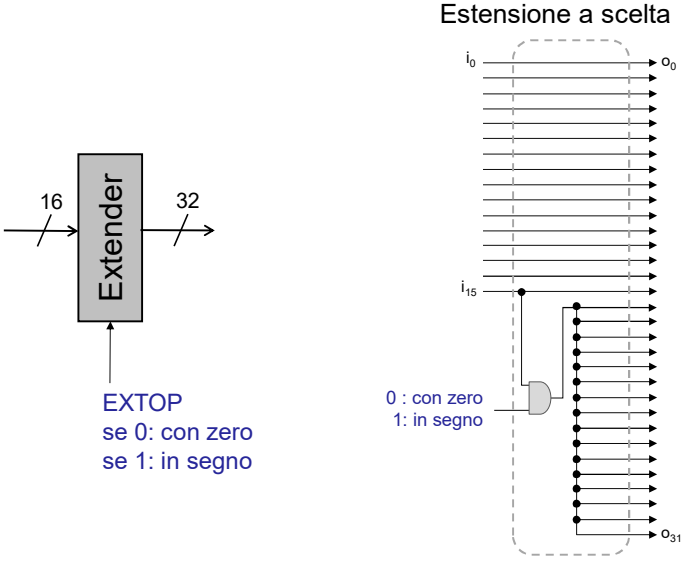
- Su 4 bit:
  1101
  denota tredici;
- allora su 8 bit
  00001101
  denota ancora tredici
  (banale)
- **Basta aggiungere tanti 0 a sx**

In complemento a 2

- Su 4 bit,
  per i positivi: 0101
  denota cinque;
- allora su 8 bit: 00000101
  denota ancora cinque
- per i negativi: 1101
  denota meno tre;
  allora su 8 bit: 11111101
  denota ancora meno tre
- **Basta ripetere il MSB a sx**

15

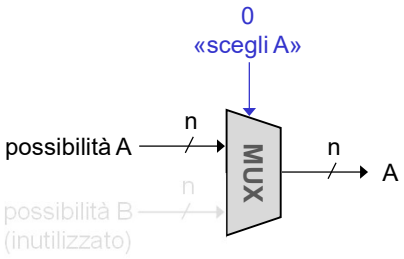
Blocco che useremo (ripasso)
Extender da 16 a 32 bit con modo a scelta



16

Blocco di cui possiamo fare uso (ripasso): Multiplexer a due vie

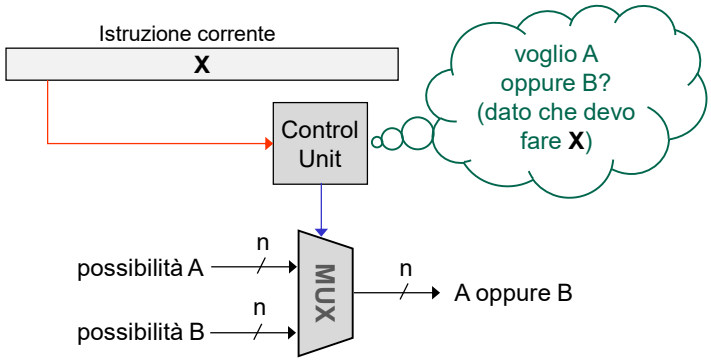
- necessari in tutti i casi è in cui è necessario fare delle scelte
- Nota:
  è del tutto influente cosa entra dai cavi della possibilità NON scelta
  - possiamo comodamente disinteressarcene
  - nei disegni che useremo, verranno a volte mostrati in grigio chiaro



17

Blocco di cui possiamo fare uso (ripasso): Multiplexer a due vie

- necessari in tutti i casi è in cui è necessario fare delle scelte
- Il data path pilotati da segnali di controllo provenienti dalla Control Unit



18