

MIPS

- Architettura RISC
- Architettura molto regolare con insieme di istruzioni semplice e compatto
- Architettura progettata per una implementazione efficiente di pipeline (lo vedremo più avanti)
- Codifica delle istruzioni omogenea: 32 bit
- Co-processore per istruzioni a virgola mobile e gestione delle eccezioni

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 65

MIPS

Registri

- 32 registri di 32 bit (registro 0 contiene sempre il valore 0)
- Architettura Load / Store
 - Istruzioni di trasferimento per muovere i dati tra memoria e registri
 - Istruzioni per la manipolazione di dati operano sui valori dei registri
 - Nessuna operazione memoria ↔ memoria
- Quindi: le istruzioni operano su registri (registro i riferito con \$i)
- Esempio: add \$0, \$1, \$2

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 66

MIPS

Dati e modi di indirizzamento

- Registri possono essere caricati con byte, mezze parole, e parole (riempiendo con 0 quando necessario o estendendo, cioè replicando, il segno sui bit non coinvolti del registro)
- Modalità di indirizzamento ammesse (con campi di 16 bit):
 - Immediata es. add \$2, \$2, 0004
 - Displacement es. sw \$1, 000c(\$1)
- Altre modalità derivabili:
 - Indiretta (displacement a 0) es. sw \$2, 0000(\$3)
 - Assoluta (registro 0 come registro base) es. lw \$1, 00c4(\$0)

Lezione 5

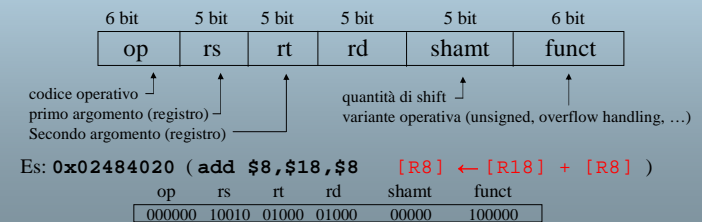
Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 67

MIPS

Formato Istruzioni

- 32 bit per tutte, 3 formati diversi (formato R, formato I, formato J)
- **Formato R (registro)**



Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 68

MIPS

Formato Istruzioni

- **Formato I (istruzioni load / store)**

6 bit	5 bit	5 bit	16 bit
op	rs	rt	address

codice operativo ↑
 primo argomento (registro) ↑
 secondo argomento (registro) ↑ displacement ↑

Es: `0x8D2804B0` (`lw $8, 1200($9) [R8] ← Mem[1200+[R9]]`)

op	rs	rt	address
100011	01001	01000	0000 0100 1011 0000

Lezione 5
Pagina 69
Architettura degli Elaboratori - 1 - A. Sperduti

MIPS

Formato Istruzioni

- **Formato J (istruzioni jump)**

6 bit	26 bit
op	address

codice operativo ↑ indirizzo ↑

Es: `0x0800AFFE` (`j 45054 [IP] ← 45054`)

op	address
000010	00 0000 0000 1010 1111 1111 1110

Lezione 5
Pagina 70
Architettura degli Elaboratori - 1 - A. Sperduti

Central Processing Unit

La CPU

- Logica operativa (data path)
- Registri
- Circuiti aritmetici dedicati
- Arithmetic Logic Unit, ALU
- Bus, instradatori e buffer
- Logica di controllo (control path)
- Set di istruzioni

Lezione 5
Pagina 71
Architettura degli Elaboratori - 1 - A. Sperduti

Unità centrale

La CPU è suddivisibile in due blocchi funzionali:

- **Sottosistema gestione dati:** *data path* (logica operativa), adibito alla memorizzazione, all'elaborazione ed all'analisi dei dati interni alla CPU
- **Sottosistema gestione istruzioni:** *control path* (logica di controllo), che ha il compito di decodificare ed interpretare le istruzioni, e di far eseguire alla logica operativa le operazioni necessarie per la loro esecuzione

Lezione 5
Pagina 72
Architettura degli Elaboratori - 1 - A. Sperduti

Parte operativa (data path)

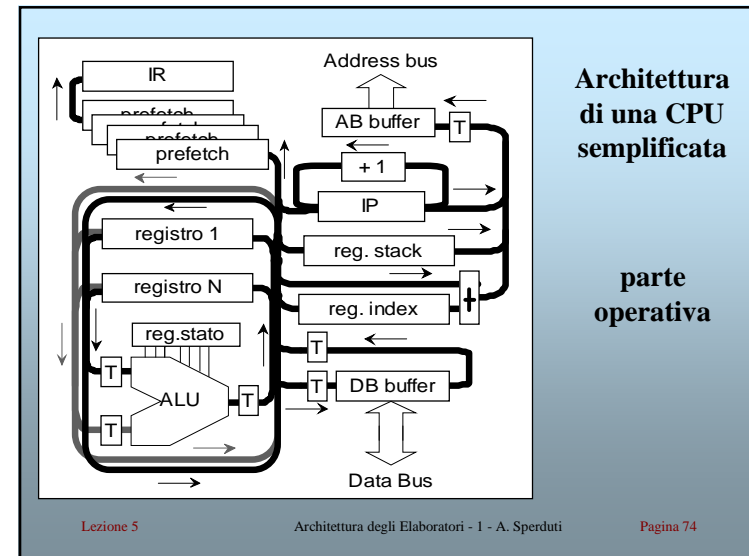
La parte operativa è costituita da vari elementi:

- Registri
- Circuiti combinatori elementari
- Arithmetic Logic Unit
- Aree di memoria di lavoro (buffer)
- Intradatori (multiplexer e demultiplexer)

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 73



Architettura di una CPU semplificata

parte operativa

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 74

Registri

- Piccole memorie temporanee molto veloci basate su componenti statici (*flip flop*)
- In numero e dimensioni variabili
- Un maggior numero di registri permette di:
 - Ridurre il numero di accessi alla memoria esterna
 - Esecuzione più veloce
 - Ridurre la complessità operativa delle istruzioni
 - Compilatore più semplice

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 75

Registri (*segue*)

I registri possono avere ciascuno una funzione specifica o essere di utilizzo generale:

- **Registri dedicati** (p.es., accumulatore): ad uso limitato ma con prestazioni migliori
- **Registri generali**: semplificano la parte controllo e l'ottimizzazione da parte dei compilatori; facilmente scalabili

La soluzione più conveniente è quella **mista**

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 76

Registri (segue)

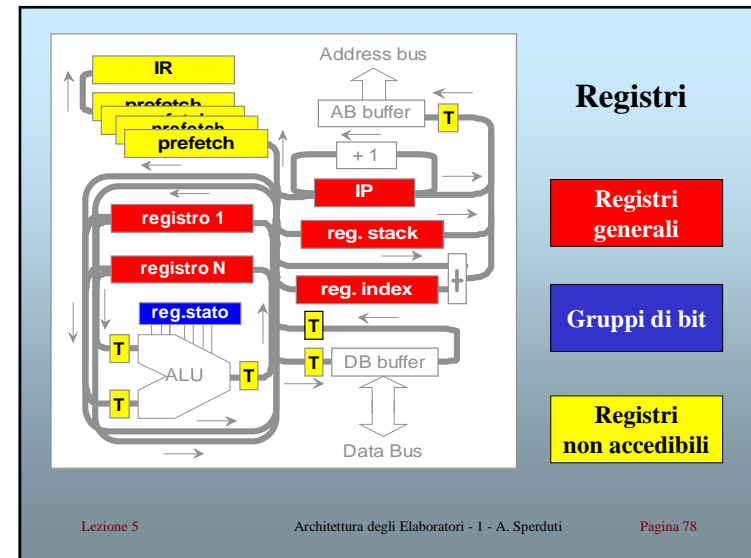
I registri **generali**

- A disposizione del programmatore
- A dimensione variabile 8, 16, 32, 64 bit
- Utilizzati per la memorizzazione dati, per il calcolo di indirizzi, per operazioni aritmetico-logiche
- Possono avere compiti specifici

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 77



Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 78

Circuiti aritmetici dedicati

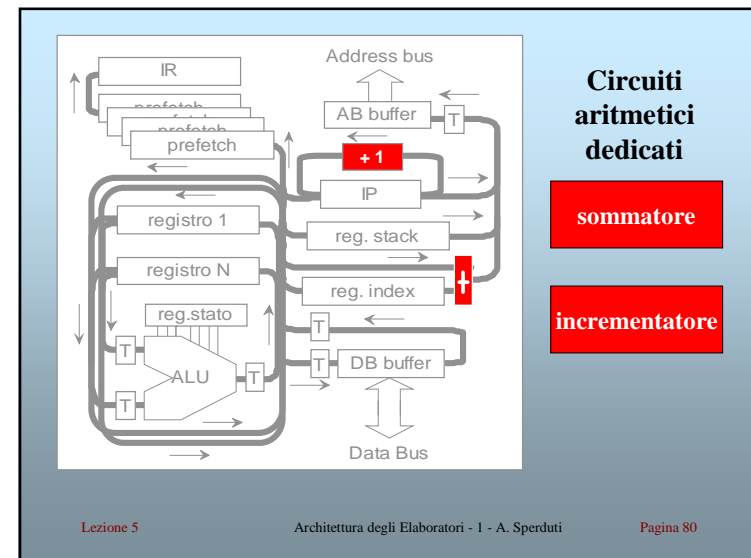
Finalizzati esclusivamente a permettere una maggiore velocità dell'ALU, p.es.:

- **Incrementatore**
 - Per far avanzare l'Instruction Pointer
 - Per particolari istruzioni auto-aggiornanti
- **Sommatore**
 - Per calcolare l'indirizzo di un operando

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 79



Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 80

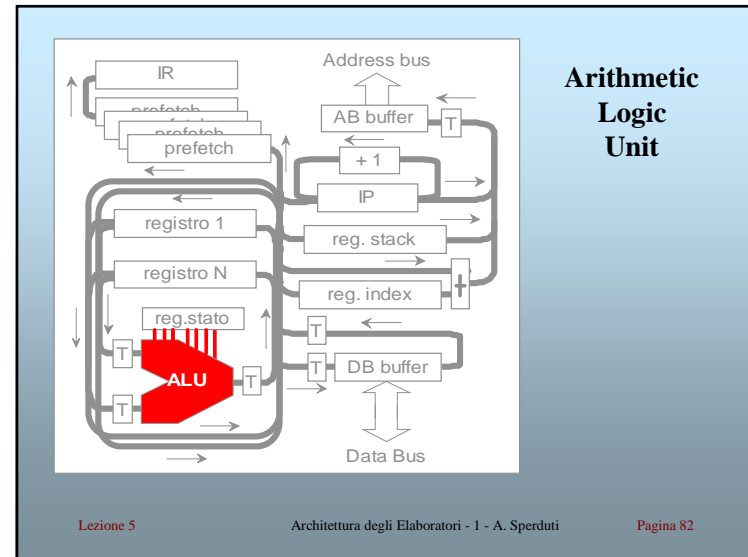
Arithmetic Logic Unit

- **Rete combinatoria multi-funzione**, in grado di effettuare operazioni aritmetiche e/o logiche su uno o due operandi
- Acquisisce ed aggiorna una serie di bit di stato, detti '**flag**', p.es.: zero, carry, overflow, segno, ... (*controllo residuo*)
- Esegue operazioni di spostamento di bit, con shifter dedicati

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 81



Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 82

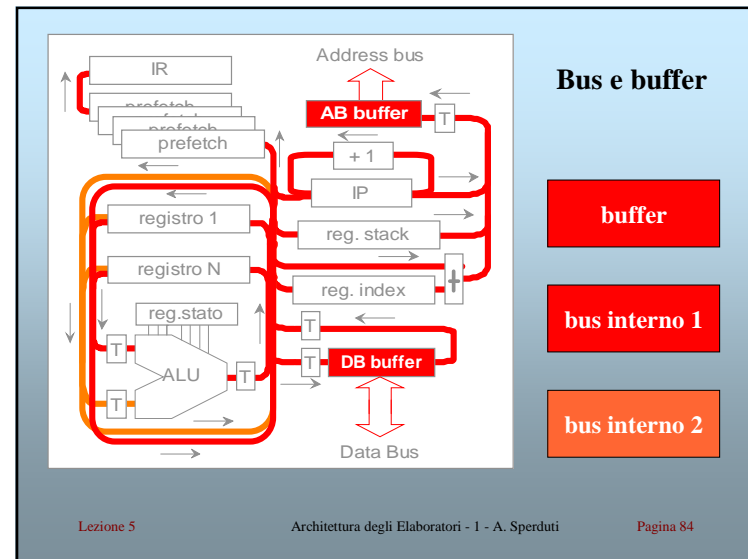
Bus, instradatori e buffer

- Le tecniche di trasferimento dati all'interno della CPU hanno grande influenza sulle prestazioni generali
 - *Efficienza*: ~ numero di cicli per istruzione; grado di parallelismo interno
- Almeno 3 architetture di trasferimento:
 - Punto-a-punto, bus singolo, bus multiplo

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 83



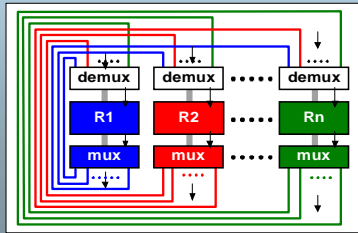
Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 84

Bus, instradatori e buffer Modalità punto-a-punto

- Un collegamento tra tutti i possibili percorsi



- Complessità di sincronizzazione
- Massima flessibilità
- Grande complessità realizzativa

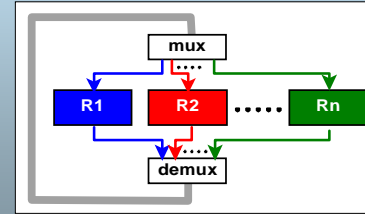
Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 85

Bus, instradatori e buffer Modalità a bus singolo

- Un solo percorso, condiviso temporalmente



- Minore complessità
- Minore flessibilità
- Più fasi operative
- Possibilità di broadcast

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 86

Bus, instradatori e buffer Modalità a bus multipli

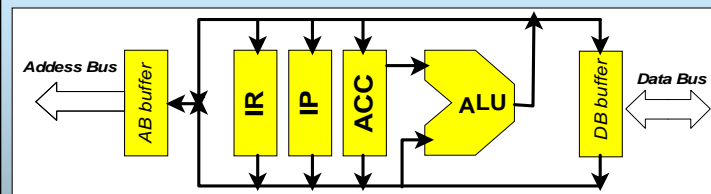
- Compromesso tra le soluzioni punto-a-punto e bus singolo
- Molteplici percorsi, sia generali che dedicati
- Maggior grado di parallelismo interno
- Il numero ottimale di bus dipende dal tipo di istruzioni e di indirizzamenti desiderati

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 87

Bus, instradatori e buffer Esempio a bus singolo



Fase di esecuzione dell'istruzione **ADD ACC, Mem[indirizzo]**

ciclo 1 IP ⇒ Internal Bus ⇒ AB buffer

...

ciclo j lettura di Memoria all'indirizzo specificato

ciclo j+1 DB buffer ⇒ Internal Bus ⇒ ALU_{op2}, ACC ⇒ ALU_{op1}, **ADD**

ciclo j+2 ALU_{res} ⇒ Internal Bus ⇒ ACC

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 88

Bus, instradatori e buffer Esempio a bus multiplo

The diagram illustrates a multi-bus architecture. It features an **Address Bus** connected to an **AB buffer**. The **Internal Address Bus** connects to the **IR** (Instruction Register) and **IP** (Instruction Pointer) registers. The **Internal Result Bus** connects to the **ACC** (Accumulator) and the **ALU** (Arithmetic Logic Unit). The **Internal Memory Bus** connects to the **DB buffer** (Data Buffer) and the **ALU**. The **Data Bus** is connected to the **DB buffer**. Arrows indicate the direction of data flow between these components.

ciclo 1 IP ⇒ Internal AB ⇒ AB buffer
 ...
ciclo j lettura di Memoria all'indirizzo specificato
ciclo j+1 DB buffer ⇒ Internal MB ⇒ ALU_{op2}, ACC ⇒ ALU_{op1},
 ADD, ALU_{res} ⇒ Internal RB ⇒ ACC

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 89

Bus, instradatori e buffer Multiplexer e demultiplexer

The diagram shows a bus system using a **demux** (demultiplexer) and a **mux** (multiplexer). The **demux** has two outputs, **A** and **B**, controlled by **IPOUT** (0=A, 1=B). The **mux** has two inputs, **A** and **B**, controlled by **IPIN** (0=A, 1=B). The bus connects to an **IP** register, an **inc** (increment) block, and a **Temp** register. Arrows show the data flow through these components.

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 90

Parte controllo (*control path*)

- Implementazione hardware (cablata) o 'microprogrammata'
- Nel caso di microprogrammazione, ad ogni **istruzione** della CPU corrisponde un **microprogramma** memorizzato in una memoria di controllo (*equivalente, ma non uguale, alla ROM*)
- L'uso di microprogrammazione **facilita** la progettazione e l'ottimizzazione della CPU
- L'esecuzione via microistruzioni è generalmente più **lenta** dell'implementazione hardware

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 91

CPU: registro Instruction Pointer (**IP**); registro generale (**AL**, 8 bit); registro indice (**SI**, 16 bit); registri interni (**IR,T**); ALU

RAM: capacità di 3072 celle di memoria da 1 byte ciascuna, dall'indirizzo 0400_{hex} a 0FFF_{hex}

ROM: capacità di 768 celle di memoria da 1 byte, da 0100_{hex} a 03FF_{hex}

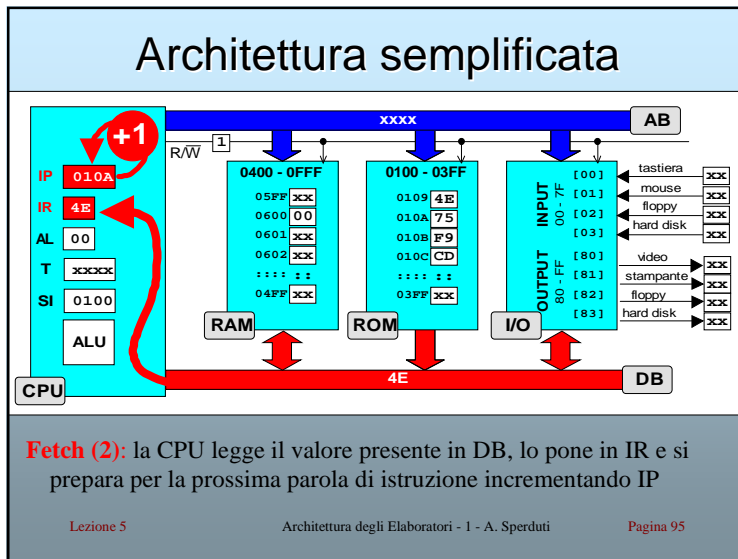
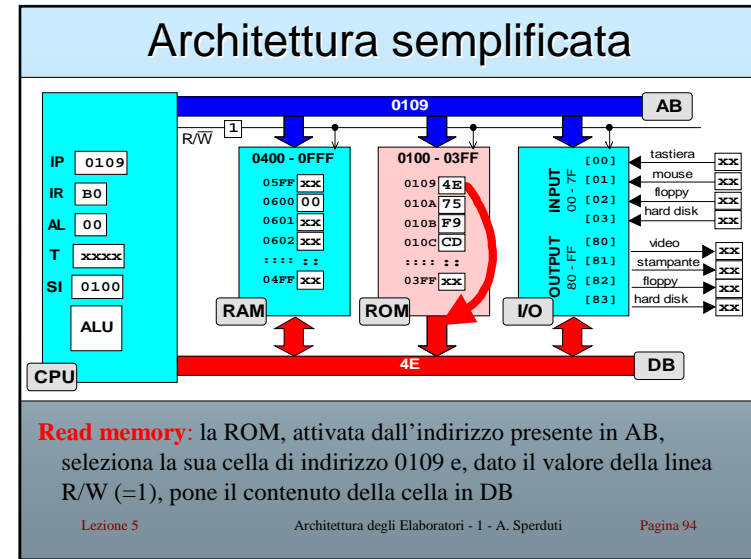
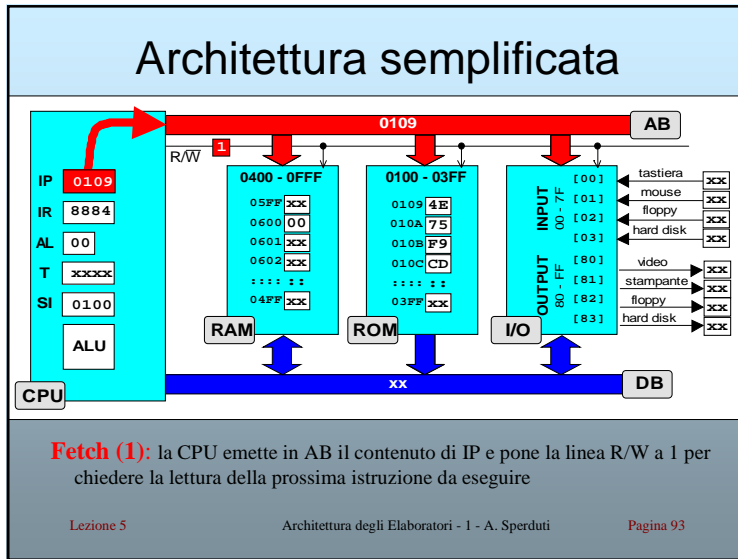
I/O: 128 indirizzi di input, da 00_{hex} a 7F_{hex}, e 128 indirizzi di output, da 80_{hex} a FF_{hex}

Esempio:

0100	BE 00 01	MOV SI,0100
0103	B0 00	MOV AL,00
0105	88 84 00 05	MOV [SI+0500],AL
0109	4E	DEC SI
010A	75 F9	JNZ 0105
010C	CD 20	INT 20

Mediante addizione in complemento a 2

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 92



Architettura semplificata

Decodifica: il codice operativo dell'istruzione corrente (4E) informa la CPU che essa deve decrementare di 1 il contenuto di SI. L'istruzione è lunga un solo byte, che ne rappresenta il **codice operativo**.

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 96

Architettura semplificata

Execute (1): la CPU invia all'ALU il contenuto di SI (0100) ed impone l'operazione di decremento di un'unità. Il risultato (00FF) viene salvato nello stesso registro SI.

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 97

Architettura semplificata

L'istruzione **DEC SI** è stata completata mediante :

- **Fetch (fase 1):** emetti IP in AB
 - *Richiesta di lettura in memoria*
- **Fetch (fase 2):** scrivi in IR il valore fornito da DB, incrementa IP, decodifica l'istruzione in IR
- **Execute (fase 1):** decrementa SI
 - *In questo caso non sono necessarie altre sottofasi*

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 98

Architettura semplificata

Fetch (1): la CPU emette in AB il contenuto di IP e pone la linea R/W a 1 per leggere la prossima istruzione da eseguire

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 99

Architettura semplificata

Read memory: la ROM, attivata dall'indirizzo presente in AB, seleziona la sua cella di indirizzo 010A e, dato il valore della linea R/W (=1), pone il contenuto della cella in DB

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 100

Architettura semplificata

Fetch (2): la CPU legge il valore presente in DB, lo pone in IR e si prepara per la prossima parola di istruzione incrementando IP

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 101

Architettura semplificata

Decodifica: il codice operativo dell'istruzione corrente (75) informa la CPU che l'istruzione è condizionata dal valore del bit di zero (**flag Z**).

Z=1 → prosegui normalmente (*jump if not zero*)

Z=0 → somma l'operando (*displacement*) ad IP

L'operazione si può effettuare su 16 bit in complemento a 2 (oppure sugli 8 bit della parte bassa dei dati trascurando l'eventuale riporto)

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 102

Architettura semplificata

Execute (1): l'istruzione ha 1 operando, dunque la CPU emette il contenuto di IP in AB e pone la linea R/W a 1, per leggere l'operando (displacement)

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 103

Architettura semplificata

Read memory: la ROM, attivata dall'indirizzo presente in AB, seleziona la sua cella di indirizzo 010B e, dato il valore della linea R/W (=1), pone il contenuto della cella in DB

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 104

Architettura semplificata

Execute (2): la CPU legge il valore presente in DB, lo pone nella parte bassa del registro T (*dei valori temporanei*), e si predispone ad acquisire la prossima parola di istruzione incrementando IP

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 105

Architettura semplificata

Execute (3a): per Z=0, la CPU richiederebbe all'ALU la somma tra (la parte bassa di) IP e (quella di) T

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 106

Architettura semplificata

Execute (3b): la CPU poi scriverebbe il risultato nel (la parte bassa del) registro IP (trascurando eventuale riporti), determinando così l'indirizzo non consecutivo dell'istruzione successiva (JUMP, salto)

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 107

Architettura semplificata

L'istruzione **JNZ F9** è stata completata mediante:

- **Fetch (1):** emetti IP in AB
 - *Richiesta di lettura in memoria*
- **Fetch(2):** scrivi in IR il valore in DB, incrementa IP, effettua decodifica
- **Execute (1):** emetti IP in AB
 - *Richiesta di lettura in memoria*
- **Execute (2):** scrivi in T il valore in DB, incrementa IP
- **Execute (3):** verifica Z, se Z=0 allora IP + T → IP

Lezione 5 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 108

Altro Esempio

Architettura con accumulatore (Assembler 6502):

- il programma (notazione esadecimale: (hex)adecimal)

INDIRIZZI	CONTENUTO	PROGRAMMA	COMMENTO
0100	9A 23	LDA #23	carica il byte con valore 23 _{hex} nell'accumulatore
0102	8A 43 20	ADC \$2043	aggiungi il byte contenuto all'indirizzo 2043 _{hex} , e se la flag di riporto è on, aggiungi 1
0105	84 44 20	STA \$2044	memorizza il risultato all'indirizzo 2044h

- la memoria (byte per byte): memorizzazione dati Little Endian

INDIRIZZI	CONTENUTO
00FF	...
0100	9A
0101	23
0102	8A
0103	43
0104	20
0105	84
0106	44
0107	20
0108	...

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 109

Altro Esempio

Sequenza fetch/execute:

- IP = 0100: Pone il contenuto di IP nell'address bus e la linea R/W ad 1. Carica il codice operativo (opcode) 9A_{hex} nel registro dati dal data bus. Incrementa IP (IP=0101). Muove il dato dal registro dati al registro IR. Decodifica il codice operativo (la decodifica segnala che è necessario un operando).
- IP = 0101: Pone il contenuto di IP nell'address bus e la linea R/W ad 1. Carica il byte = 23_{hex} nel registro dati. Incrementa IP (IP=0102; punta alla istruzione ADC \$2043). Muove il dato nell'accumulatore ACC.
- IP = 0102: Pone il contenuto di IP nell'address bus e la linea R/W ad 1. Carica il codice operativo 8A_{hex} nel registro dati dal data bus. Incrementa IP (IP=0103). Muove il dato dal registro dati al registro IR. Decodifica il codice operativo (la decodifica segnala che i prossimi 2 byte in memoria contengono l'indirizzo del dato da utilizzare).

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 110

Altro Esempio

Sequenza fetch/execute:

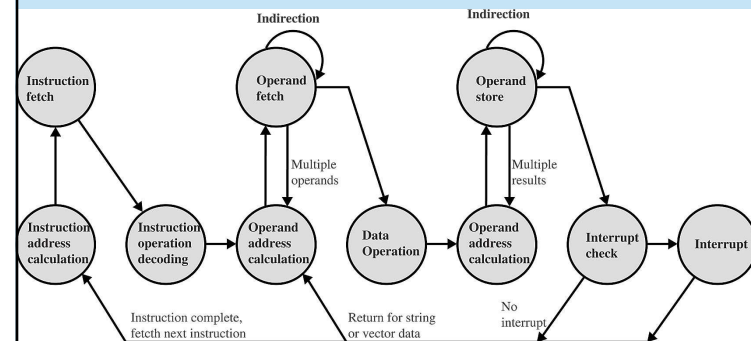
- IP = 0103: Pone il contenuto di IP nell'address bus e la linea R/W ad 1. Carica il byte = 43_{hex} nel registro dati. Incrementa IP (IP=0104). Muove il dato dal registro dati al byte meno significativo di un registro temporaneo T (di 2 byte).
- IP = 0104: Pone il contenuto di IP nell'address bus e la linea R/W ad 1. Carica il byte = 20_{hex} nel registro dati. Incrementa IP (IP=0105). Muove il dato dal registro dati al byte piu' significativo del registro temporaneo T. Muove il contenuto del registro temporaneo nell'address bus e pone la linea R/W ad 1. Carica il byte contenuto nella locazione 2043_{hex} nel registro dati. Somma il dato ad ACC. Setta flag di riporto (carry) se la somma genera overflow.

Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 111

Fetch/Execute: ancora più in dettaglio



Lezione 5

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 112