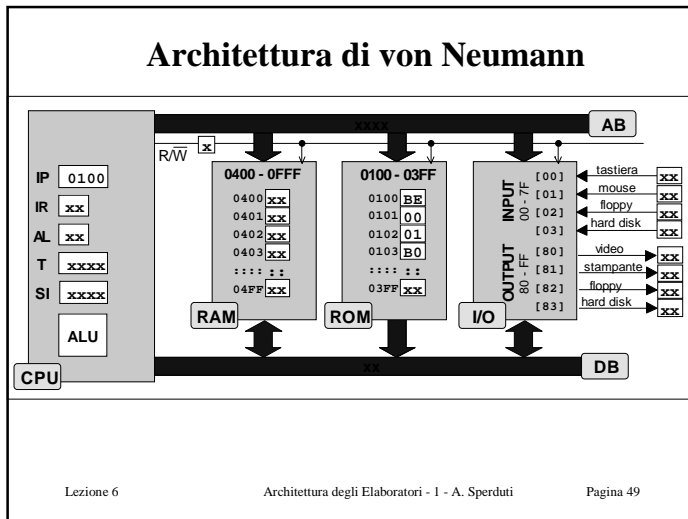
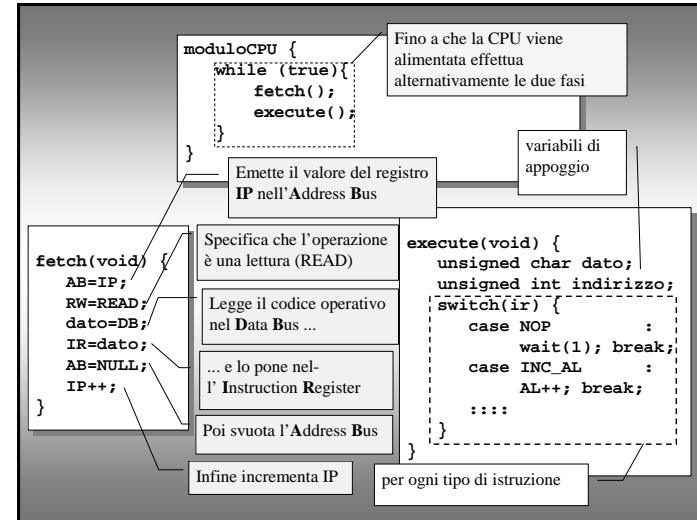


Central Processing Unit - *indice*

- 2 La CPU
 - 2.1 Logica operativa (data path)
 - 2.2 Registri
 - 2.3 Circuiti aritmetici dedicati
 - 2.4 Arithmetic Logic Unit, ALU
 - 2.5 Bus, instradatori e buffer
 - 2.6 Logica di controllo (control path)
 - 2.7 Set di istruzioni

Lezione 6 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 48



2. 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 operative le operazioni necessarie per la loro esecuzione

Lezione 6 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 51

2.1 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 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 52

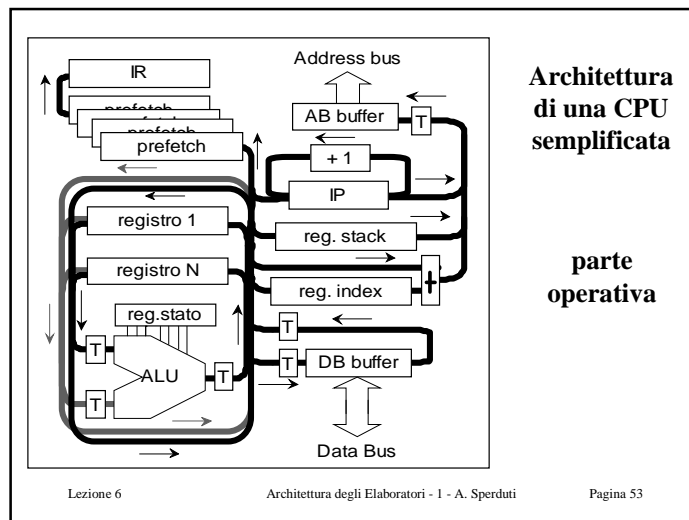
2.2 Registri

- 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 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 54



2.2 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 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 55

2.2 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 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 56

2.3 Circuiti aritmetici dedicati

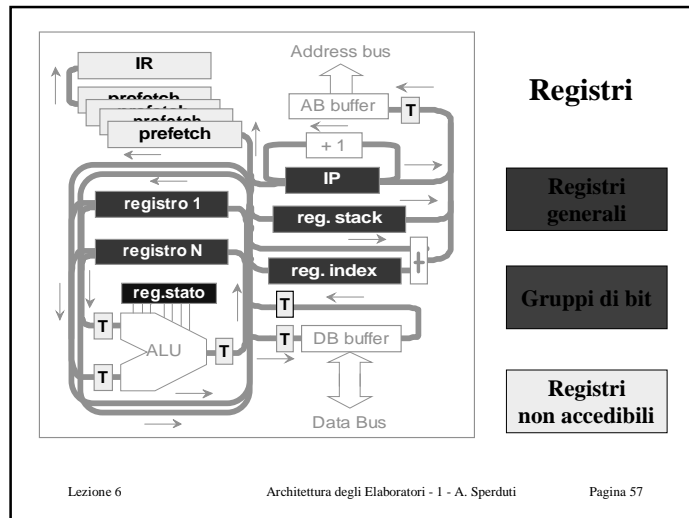
Finalizzati esclusivamente a permettere una maggiore velocità dell'ALU, p.es.:

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

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

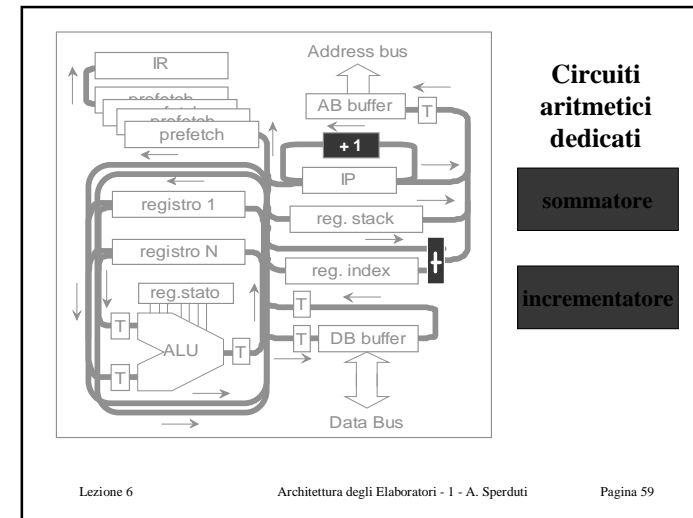
Pagina 58



Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 57



Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 59

2.4 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 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 60

2.4 Arithmetic Logic Unit

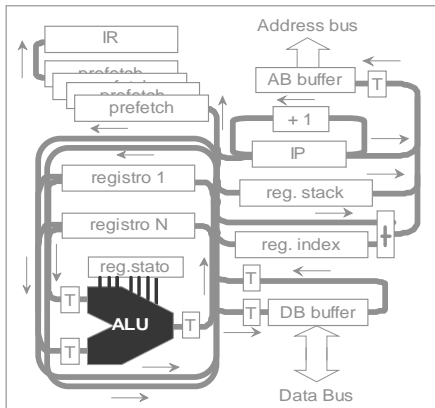
Codifica dei dati interi senza segno

TIPO	bit	rango
UB unsigned byte	8	0 .. 255
UW unsigned word	16	0 .. 65525
UD unsigned double word	32	0 .. ~ 4GB
BU byte unpacked BCD	8	0 .. 9
BP byte packed BCD	8	0 .. 99

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 62



Arithmetic Logic Unit

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 61

2.4 Arithmetic Logic Unit

Codifiche UB, UW, UD

Dato il numero ad **n** bit

$$A = a_{n-1} a_{n-2} \dots a_2 a_1 a_0$$

la *i*-esima cifra ha un peso di 2^i , ad esempio

$$1010_{\text{bin}} = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 8 + 2 = 10_{\text{dec}}$$

L'intervallo di valori va da 0 a $2^n - 1$

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 63

2.5 Arithmetic Logic Unit

Codifica dei numeri in virgola mobile (2)

Lo standard ANSI/IEEE 754 usa la rappresentazione $(-1)^S \times 1.M \times 2^{E-bias}$ a 3 campi:

- **S**, *segno*, su 1 bit (0 → numero positivo)
- **M**, *mantissa*, su **m** bit, normalizzata escludendo il bit più significativo che vale sempre 1
- **E**, *esponente o caratteristica*, su **e** bit, valutata in codifica ad eccesso $2^{e-1} - 1$ (**bias**)
- Con precisione **p = 1 + m + e**

Lezione 6 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 76

2.5 Arithmetic Logic Unit

Codifica dei numeri in virgola mobile (4)

Esempio
Il valore in codifica SR (*single-precision*)

0 10000110 000000010000000000000000
2⁻¹ 2⁻² 2⁻³ 2⁻⁴ ...

dove **S** = 0, **E** = 10000110 = 134_{dec}, e
M = 000000010₂.0_{IEEE754} = 00000001₂/10000000₂ = 0.00390625
 (equivalente a 2⁻⁸, con le cifre in mantissa aventi peso negativo)
 corrisponde al valore decimale:
 $x = (-1)^0 \times 1.00390625 \times 2^{134-127} = 1.00390625 \times 128 = 128.5$

Lezione 6 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 78

2.5 Arithmetic Logic Unit

Codifica dei numeri in virgola mobile (3)

TIPO	p	S	e	m	bias
SR short real	32	1	8	23	127
	$\sim 1.8 \times 10^{-38} < x < \sim 3.40 \times 10^{+38}$				
LR long real	64	1	11	52	1023
	$\sim 2.23 \times 10^{-308} < x < \sim 1.80 \times 10^{+308}$				
XR extended real	80	1	15	64	16383
	$\sim 3.4 \times 10^{-4932} < x < \sim 1.2 \times 10^{+4932}$				

Lezione 6 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 77

2.5 Arithmetic Logic Unit

Somma di numeri in virgola mobile

Dati $X = X_s \times X_{m_x} \times 2^{e_x}$ e $Y = Y_s \times Y_{m_y} \times 2^{e_y}$ ($|X| < |Y|$)

- Calcolare $n = e_y - e_x$
- Spostare X_{m_x} n volte a destra, ottenendo $X'_m = X_{m_x} / 2^n$
- Calcolare $Z_m = X'_m + Y_{m_y}$ e normalizzare ad m bit
- Fissare $e_z = e_y$
- Calcolare Z_s (il segno del risultato)
- Il risultato vale $X + Y = Z_s \times Z_m \times 2^{e_z}$

Lezione 6 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 79

2.5 Arithmetic Logic Unit

Memorizzazione dei dati Little Endian

- **Little Endian:** si memorizzano i byte meno significativi dell'informazione a partire dagli indirizzi inferiori
- L'indirizzo del dato è quello dell'LSB (*small end*)

	0122	
MOV SI,0123	0123	AB
MOV [SI],89AB	0124	89
	0125	

Intel x86
DEC Vax

Lezione 6
Architettura degli Elaboratori - 1 - A. Sperduti
Pagina 84

Bus e buffer

Lezione 6
Architettura degli Elaboratori - 1 - A. Sperduti
Pagina 86

2.6 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 6
Architettura degli Elaboratori - 1 - A. Sperduti
Pagina 85

2.6 Bus, instradatori e buffer

Modalità punto-a-punto

- Un collegamento tra tutti i possibili percorsi

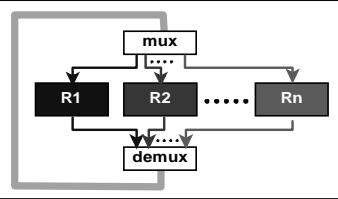
- Complessità di sincronizzazione
- Massima flessibilità
- Grande complessità realizzativa

Lezione 6
Architettura degli Elaboratori - 1 - A. Sperduti
Pagina 87

2.6 Bus, instradatori e buffer

Modalità a bus singolo

- Un solo percorso, condiviso temporalmente

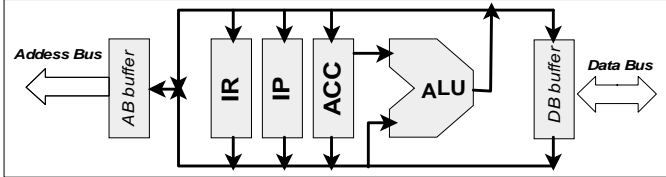


- Minore complessità
- Minore flessibilità
- Più fasi operative
- Possibilità di broadcast

Lezione 6
Architettura degli Elaboratori - 1 - A. Sperduti
Pagina 88

2.6 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 6
Architettura degli Elaboratori - 1 - A. Sperduti
Pagina 90

2.6 Bus, instradatori e buffer

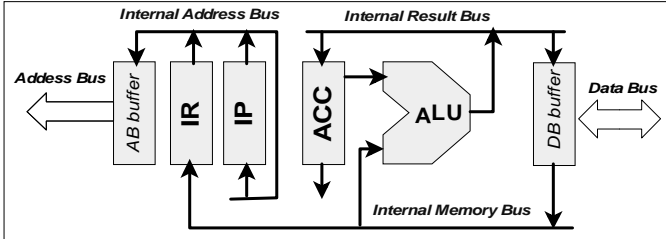
Modalità a bus multipli

- 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 6
Architettura degli Elaboratori - 1 - A. Sperduti
Pagina 89

2.6 Bus, instradatori e buffer

Esempio a bus multiplo



ciclo 1 IP ⇒ Internal AB ⇒ AB buffer

...

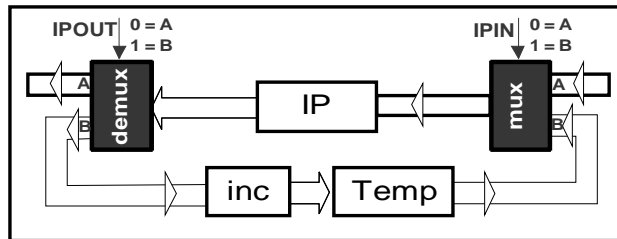
ciclo j lettura di Memoria all'indirizzo specificato

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

Lezione 6
Architettura degli Elaboratori - 1 - A. Sperduti
Pagina 91

2.6 Bus, instradatori e buffer

Multiplexer e demultiplexer



Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 92

2.7 Instruction set 1

- L'istruzione set specifica le **operazioni esterne** della CPU
- Le istruzioni possono prevedere **operandi** con varie modalità di localizzazione
- Le istruzioni possono avere **vari formati** e **diversa durata**
- Il tipo di istruzioni previsto da un instruction set determina il costo e le prestazioni della CPU

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 94

2.6 Parte controllo (*control path*)

- Già discussa come 'PC microprogrammata'
- 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 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 93

2.7 Instruction set 2

- Un instruction set definisce (corrisponde ad) una particolare **architettura** di CPU
 - Architettura a pila (*stack*)
 - Architettura ad accumulatore
 - Architettura a registri generali

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 95

2.7 Architettura a pila

- Gli operandi sono assunti essere presenti **in testa** ad una particolare zona di memoria detta **'pila di sistema'**
- Le operazioni di calcolo sono precedute da inserimento di operandi (**PUSH**) in pila e seguite da prelievo del risultato (**POP**) dalla pila
- Architettura arcaica e poco efficace
- Programmazione verbosa

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 96

2.7 Architettura a registri generali

- Gli operandi sono tutti **espliciti**
- Gli operandi sono localizzati in **memoria** o in **registri generali**
- Esiste una vasta gamma di modalità di localizzazione (**indirizzamento**) degli operandi

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 98

2.7 Architettura ad accumulatore

- Uno degli operandi dell'istruzione è assunto trovarsi sempre in un registro speciale detto **'accumulatore'**
- Gli altri operandi sono generalmente localizzati in memoria
- Architettura arcaica e poco efficace
- Programmazione concisa

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 97

2.7 Esempio

A + B → C

Architettura a pila

Push A | Push B | Add | Pop C

A → Stack | B → Stack | <risultato> → Stack | Stack → C

Architettura ad accumulatore

Load A | Add B | Store C

A → Acc | Acc + B → Acc | Acc → C

Architettura a registri generali

Load R1, A | Load R2, B | Add R1, R2, R3 | Store C, R3

A → R1 | B → R2 | R1 + R2 → R3 | R3 → C

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 99

2.7 Ulteriore classificazione 1

Esistono **tre** sottoclassi di architetture a registri generali

- **Registro-registro (Load/Store)**
Tutti gli operandi sono su registro. Gli accessi a memoria si effettuano con **Load** (prelievo) e **Store** (deposito)
- **Memoria-registro**
Prevedono operazioni con alcuni operandi in memoria ed altri su registro
- **Memoria-memoria**
Prevedono operazioni che prelevano operandi in memoria e restituiscono il risultato in memoria

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 100

2.7 Architettura Load/Store (RISC)

- Necessita di poche istruzioni, con formato molto semplice ed uniforme
- Le istruzioni hanno tempi di esecuzione molto simili tra loro
- Facilita il compito dei generatori automatici di codice macchina (compilatori)
- Nota sotto il nome di architettura **RISC** (Reduced Instruction Set Computer)

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 102

2.7 Ulteriore classificazione 2

- Architetture commerciali di tipo 1 (Load/Store)
 - SPARC, MIPS, PowerPC
- Architetture commerciali di tipo 2 (memoria/registo)
 - Intel x86, Motorola 68k
- **Non** esistono architetture commerciali di tipo 3 (memoria/memoria)

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 101

2.7 Architettura CISC

- Alternativa alla regolarità dell'architettura RISC è la cosiddetta architettura **CISC** (Complex Instruction Set Computer), che utilizza istruzioni a **formato variabile**, con grande potenza espressiva e tempi di esecuzioni variabili e spesso elevati
- L'architettura Intel x86 è di tipo CISC

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 103

2.7 Modalità di indirizzamento

Registro	Add R1,R2	$[R1] \leftarrow [R1] + [R2]$
Immediata	Add R1,K	$[R1] \leftarrow [R1] + K$
Displacement	Add R1,K,[R2]	$[R1] \leftarrow [R1] + \text{Mem}[K + [R2]]$
Indiretta	Add R1,[R2]	$[R1] \leftarrow [R1] + \text{Mem}[[R2]]$
Indicizzata	Add R1,[R2+R3]	$[R1] \leftarrow [R1] + \text{Mem}[[R2] + [R3]]$
Assoluta	Add R1,[K]	$[R1] \leftarrow [R1] + \text{Mem}[K]$
Memoria indiretta	Add R1,@[R2]	$[R1] \leftarrow [R1] + \text{Mem}[\text{Mem}[[R2]]]$

Lezione 6 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 104

2.7 Accesso a memoria

dato
(s=2, half word):
[01 | 5A]

$A = 1001_{16}$
 \uparrow
 0001_2
 $1001_{16} \bmod 2_{16} = 0001_2$

$A = 1000_{16}$
 \uparrow
 0000_2
 $1000_{16} \bmod 2_{16} = 0000_2$

Memoria

Lezione 6 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 106

2.7 Accesso a memoria

- A questo livello, la memoria viene vista come un array di byte
- Per ogni richiesta di un dato ad un certo indirizzo, la CPU ottiene un numero di byte determinato dall'ampiezza del Data Bus e dalle modalità di **allineamento** dei dati in memoria
- Un accesso ad un dato (composto da s byte) all'indirizzo A si dice **allineato** se $A \bmod s = 0$

Formato del dato	Allineato (ultimi 3 bit di A)	Non allineato
byte	0,1,2,3,4,5,6,7	mai
half word (2 byte)	0,2,4,6	1,3,5,7
word (4 byte)	0,4	1,2,3,5,6,7
double word (8 byte)	0	1,2,3,4,5,6,7

Lezione 6 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 105

2.7 Accesso a memoria

- Memoria non allineata (Intel x86)**
 - Accessi facilitati ed ottimizzati
 - Maggiore complessità di progettazione
- Memoria allineata (SPARC, Motorola 68k)**
 - Accesso più complesso a dati non allineati
 - Richiede compilatori efficienti per preservare capacità di accesso adeguata a dati non allineati

Lezione 6 Architettura degli Elaboratori - 1 - A. Sperduti Pagina 107

2.7 Tassonomia di istruzioni

<i>Tipo</i>	<i>Esempi</i>
aritmetiche e logiche	aritmetica intera e operazioni logiche: add, and, subtract, or
trasferimento dati	trasferimento dati da/per memoria/registri: load, store
controllo	branch, jump, chiamata e ritorno da procedure, traps
sistema	chiamate di sistema operativo, istruzioni per la gestione della memoria virtuale
virgola mobile	operazioni in virgola mobile: add, multiply
decimale	operazioni decimali: decimal add, decimal multiply, decimal-to-character conversions
stringa grafica	spostamento, comparazione, ricerca di stringhe operazioni su pixel, operazioni di compressione/decompressione

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 108

2.7 Altri tipi di istruzioni

- Trasferimento di controllo
 - Operanti su Instruction Pointer (IP), detto anche Program Counter (PC)
- Operazioni su stack
 - Push, Pop
- Operazione specializzate
 - Aritmetica BCD, trattamento I/O

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 110

2.7 Istruzioni più usate (80x86)

<i>istruzione</i>	<i>percentuale di uso</i>
load	22%
branch condizionale	20%
compare	16%
store	12%
add	8%
and	6%
sub	5%
spostamento reg./reg.	4%
call	1%
return	1%
Totale	96%

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 109

2.7 Formato delle istruzioni

- Come vengono codificate in binario le istruzioni ?
- Bisogna specificare:
 - di quale operazione si tratta (**opcode**);
 - il tipo/i di indirizzamento di memoria usato/i (**address mode(s)**)
- Formati:
 - variabile

op. e #operandi	addr. specifier 1	...	addr. specifier n
-----------------	-------------------	-----	-------------------
 - fisso

op.	addr. field 1	addr. field 2	addr. field 3
-----	---------------	---------------	---------------
 - ibrido

op.	addr. field 1	addr. field 2
op.	addr. field 1	addr. field 2
...		

 stessa istruzione a lunghezza multipla, ma ognuna in formato fisso

Lezione 6

Architettura degli Elaboratori - 1 - A. Sperduti

Pagina 111

2.7 Formato delle istruzioni

- Esempi:

- variabile (VAX):

```
add13 r1,737(r2),(r3)
```

- fisso (DLX)

```
add r1,r2,r3
```

- ibrido (80x86)

```
fadd          operandi e risultato in stack
```

```
fadd st(i)    un operando in registro i sotto la stack ...
```

```
fadd mem32    un operando è una cella di memoria a 32 bit ...
```