Classe 3-I - Matricole 4,5
Lezione #5

Laboratorio di Informatica di Base

Ferdinando Cicalese

cicalese@dia.unisa.it

Il linguaggio del calcolatore

- Consideriamo la categoria dei linguaggi macchina (direttamente eseguibili)
- Modello operazionale (eseguire passo passo un programma)
  - Formato delle istruzioni
  - Esecuzione delle istruzioni
  - Principali istruzioni
  - Rappresentazione dei dati in memoria
  - Modi di indirizzamento
  - Linguaggio assemblatore

Formato delle istruzioni

- Istruzioni, memorizzate in memoria, consistono:
  - Codice operativo (sempre presente)
  - Uno o piu' operandi (locazioni delle celle di memoria)
- Lunghezza delle istruzioni s = m + n bit
  - La sequenza m specifica il codice
  - La sequenza n specifica l'operando
- 2^m sono i possibili codici operativi
- n >= k perche' bisogna indirizzare tutta la memoria 2^k celle
- Se s = h allora ogni istruzione e' contenuta in una cella

Esecuzione delle istruzioni

- Tre fasi che gestiscono una sequenza di operazioni
  - Acquisizione della memoria (fetch)
  - Interpretazione
  - Esecuzione (execute)

Acquisizione della memoria (fetch)

- 4 microistruzioni (trsferimento di dati)
  - Il contenuto del PC viene trsferito in RI
  - La cella di memoria indirizzata viene letta
  - Il contenuto di RD viene trasferito in RIC
  - Il contenuto di PC = PC +1 (eccezione per il jump)

  Esecuzione in microistruzioni:

```
PC -> RI
MEM[RI] -> RD
RD -> RIC
PC+1 -> PC
```

Interpretazione ed esecuzione delle istruzioni

- Il contenuto del regidtro RIC viene interpretato
  - Viene analizzato solo il codice operativo (m bit)
- L'operazione specificata viene realizzata

Alcune delle principali istruzioni

- LOADA e LOADB (lettura della memoria)
- L'istruzione LOADA ind1

  (ind1 e' un indirizzo presente nel campo operando a n bit)

Esecuzione in microistruzioni:

```
OP(RIC) -> RI
MEM[RI] -> RD
RD -> RA(copia)
```

Alcune delle principali istruzioni

- STOREA e STOREB (scrittura della memoria)
- L'istruzione STOREA ind1

  (ind1 e' un indirizzo presente nel campo operando a n bit)

Esecuzione in microistruzioni:

```
RA -> RD(copia)
OP(RIC) -> RI
RD -> MEM[RI]
```

Alcune delle principali istruzioni

- READ e WRITE (lettura e su una periferica)
- READ ind1 (ind1 e' un indirizzo presente nel campo operando a n bit) e WRITE ind1

Esecuzione in microistruzioni:

RDP -> RD(trasf.)	OP(RIC) -> RI
OP(RIC) -> RI	MEM[RI] -> RD
RD -> MEM[RI]	RD -> RDP

Alcune delle principali istruzioni

- Operazioni tra numeri
  - ADD, DIF, MUL, DIV
- Tutte coinvolgono 2 operandi: A e B
  - A conterra' il risultato
  - In caso di DIV, B conterra' il resto

Alcune delle principali istruzioni

- L'istruzione JUMP (salto)
  - Modificano la normale esecuzione del programma
- JUMP ind1 (incondizionato)
- JUMPz ind1 (condizionato) se il registro A e' nullo

In microistruzioni:

```
ind1 -> PC
```

```
Se RS = 1
ind1 -> PC
```

Alcune delle principali istruzioni

- L'istruzione NOP (no operation)
  - Attesa di eventi esterni (si aspetta un ciclo istruzione)
- L'istruzione HALT (termina il programma)

Alcune delle principali istruzioni

- 0000 LOADA
- 0001 LOADB
- 0010 STOREA
- 0011 STOREB
- 0100 READ
- 0101 WRITE
- 0110 ADD
- 0111 DIF
- 1000 MUL
- 1001 DIV

1010 JUMP
1011 JUMPZ
1100 NOP
1101 HALT

$14 <= 2^4$ bit

Rappresentazione dei dati in memoria

- Ogni istruzione trasforma:

  Dati di input (acquisiti mediante lettura) in dati di output (calcolati e poi memorizzati mediante scrittura)

- Finita l'elaborazione i dati vengono persi

- Nel linguaggio macchina finora visto si possono manipolare solo i tipi di dati interi

Un programma per moltiplicare interi

- Consiste di istruzioni e dati memorizzati in due separate aree della memoria...

- 0 READ 8
- 1 READ 9
- 2 LOADA 8
- 3 LOADB 9
- 4 MUL
- 5 STOREA 8
- 6 WRITE 8
- 7 HALT
- 8 int
- 9 int

- 0100000000001000
- 0100000000001001
- 0000000000001000
- 0001000000001000
- 1000000000000000
- 0010000000001000
- 0101000000001000
- 1101000000000000
- 0000000000000000
- 0000000000000000

Un programma per moltiplicare interi

- In microistruzioni...

0 : fetch
PC -> RI
MEM[RI] -> RD
RD -> RIC
PC +1 -> PC
0: Esecuzione
RDP -> RD
OP[RIC] -> RI
RD -> MEM[RI]
.....

Modi di Indirizzamento dell'operando

- Diretto (indirizzo assoluto)
- Indiretto (indirizzo dell' indirizzo)
- Relativo a un registro indice (registro indice + indirizzo operando)
- Immediato (no indirizzo)

Modi di Indirizzamento dell'operando

Diretto: indirizzo assoluto della parola di memoria in cui e' memorizzato l'operando

- Richiede alla macchina un accesso alla memoria
- Presenta due inconvenienti
  - Spostare il programma significa ricalcolare gli indirizzi
  - Se le memoria e' grande si devono usare indirizzi lunghi

Modi di Indirizzamento dell'operando

Indiretto: l'indirizzo e' contenuto in mem[indirizzo]

Il vantaggio:

- l'indirizzo puo' essere di h bit (piu' grande)
- puoi spostare il programma e non i dati

...mentre richiede

- un doppio accesso alla memoria
- comunque il secondo e' un indirizzo assoluto

Modi di Indirizzamento dell'operando

Relativo a un registro indice: l'indirizzo e' calcolato registro indice (I) + indirizzo operando

- Unico accesso alla memoria
- Se si vogliono prelevare P parole di memoria che seguono una determinata parola(base) si calcola il successivo indirizzo incrementando di 1 il registro I
- Maggiore capacita' di memoria 2^h

Modi di Indirizzamento dell'operando

Immediato nesun indirizzo e' specificato

- C'e' solo il limite sul numero di bit =n campo operando

La CPU riconosce i diversi modi di indirizzamento aggiungendo all'inseme di istruzioni

Es: per LOADA

- LOADDIFA
- LOADINDA
- LOADIMMA

Inconveniente:
Il linguaggio deve contenere un numero >> di istruzioni

Modi di Indirizzamento dell'operando

- Per evitare questo carico di istruzioni gli indirizzamenti vengono CODIFICATI

OO diretto
01 differito
10 indice

- Immediato

La parola istruzione s=m+n diventa s=m+i+n

Il "livello" dei Linguaggi

- I linguaggi ad alto livello migliorano la scrittura e comprensione dei programmi
- Maggiore e' il livello di astrazione piu' complesso e' il programma (compilatore) che traduce il codice sorgente in quello eseguibile(programma oggetto)
- Il compilatore e gli interpreti sono parte dell'ambiente di programmazione
- Il linguaggio assembler e' un linguaggio a basso livello

(molto vicino al linguaggio macchina)

Il Linguaggio Assembler

Corrispondenza tra istruzioni in linguaggio macchina e in assebler

- Non e' necesssario
  - Sapere la codifica binaria
  - I riferimenti ai dati e alle istruzioni sono fatti in maniera simbolica
  - Codificare gli indirizzamenti

```
LOADA NUM
LOADA @NUM
LOADA NUM(I)
LOADA #NUM
```

NUM e' la etichetta per una cella di memoria che contiene il dato

Il Linguaggio Assembler

- Un programma in linguaggio assembler non puo' essere eseguito
- L'assemblatore trasforma keywords in codidi operativi e etichette in indirizzi
- Il caricamento del programma in memoria e' detto rilocazione ed e' effettuata dal sistema operativo