LABORATORIO DI PROGRAMMAZIONE
Corso di laurea in matematica

5 – LA RAPPRESENTAZIONE DELLE ISTRUZIONI

Marco Lapegna
Dipartimento di Matematica e Applicazioni
Universita' degli Studi di Napoli Federico II

wpage.unina.it/lapegna

1

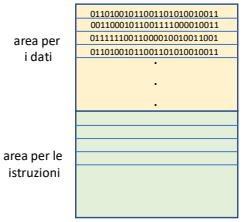
Dalla precedente lezione

Ad un programma in esecuzione viene assegnata
un'area di memoria in cui sono conservati

istruzioni e dati

Rappresentazione dei dati

- Tipo intero
- Tipo alfanumerico
- Tipo logico
- Tipo reale



Rappresentazione delle istruzioni ?

2

Il linguaggio macchina

Nelle locazioni di memoria e' possibile memorizzare solo sequenze di bit
Il linguaggio per rappresentare le istruzioni come sequenze di bit e' detto

LINGUAGGIO MACCHINA
(Instruction Set Architecture)

Attraverso il linguaggio macchina e' possibile accedere
alle **risorse fisiche** dalla macchina (CPU, memoria e unita' di I/O)

Quasi **ogni CPU ha un proprio** linguaggio macchina

Il linguaggio macchina e **l'unico linguaggio**
direttamente eseguibile dal calcolatore

3

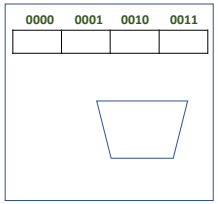
esempio

Si vuole eseguire l'operazione **3 + 4**
con i **dati** memorizzati nelle locazioni con indirizzi **0001** e **0010**
e riporre il **risultato** nella locazione con indirizzo **0100**

i dati risiedono in memoria

indirizzi	MEMORIA
0000	
0001	0000 0011
0010	0000 0100
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	

La ALU contiene appositi supporti
per contenere i dati in transito da
e per la memoria chiamati
REGISTRI



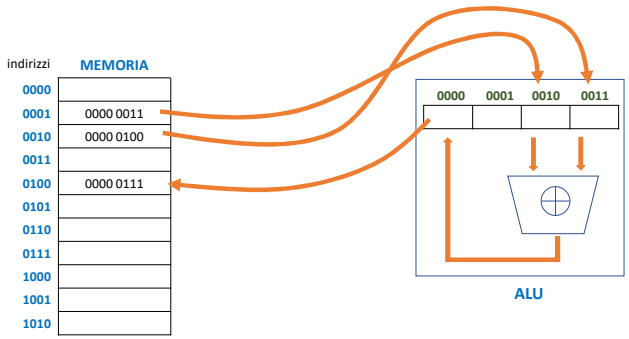
4

esempio

1) bisogna spostare i dati dalla memoria ai registri della ALU

2) bisogna eseguire la somma utilizzando i registri

3) bisogna spostare il risultato dei registri alla memoria



5

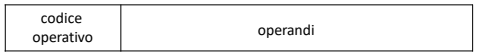
istruzioni in linguaggio macchina

Per eseguire l'operazione richiesta e' necessario:

1. **prelevare** il primo operando dalla locazione con indirizzo **0001** e lo copia in un registro della ALU (ad es. registro **0010**)
2. **prelevare** il secondo operando dalla locazione con indirizzo **0010** e lo copia in un registro della ALU (ad es. registro **0011**)
3. **sommare** i valori dei registri e riporre il risultato in un registro (ad es. **0000**)
4. **riponi** il risultato nella locazione con indirizzo **0100**

In ogni caso bisogna specificare

1. l'**azione** da compiere, attraverso un codice operativo (stringa di bit)
2. gli **operandi** su cui agire (indirizzi di memoria, registri o direttamente valori)



Esistono istruzioni con 1, 2 oppure 3 operandi

6

esempi di codici operativi

supponiamo che le operazioni siano identificate dai seguenti codici operativi

- **preleva** un dato dalla memoria c.o. = **0000**
- **riponi** un dato in memoria c.o. = **0001**
- **somma** il contenuto di due registri c.o. = **1000**

le istruzioni in linguaggio macchina per l'operazione richiesta diventano

0000 0010 0001
0000 0011 0010
1000 0000 0010 0011
0001 0000 0100

7

ciclo fech/decode/execute

Le istruzioni vengono eseguite mediante le seguenti fasi

1. lettura dalla istruzione dalla memoria — **fetch**
2. determinazione del significato del c.o. — **decode**
3. determinazione degli indirizzi degli operandi — **decode**
4. movimento dei dati dalla memoria alla ALU — **execute**
5. esecuzione dell'operazione — **execute**
6. movimento del risultato dalla ALU alla memoria — **execute**
7. determinazione dell'indirizzo della successive istruzione

Sono eseguite dall'Unita' di Controllo

8

caratteristiche del linguaggio macchina

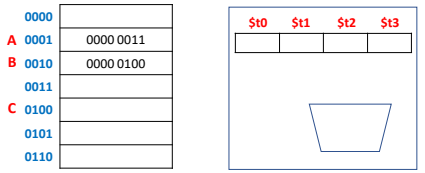
- **Non e' portabile** su calcolatori con architettura differente
- **Parecchie istruzioni** anche per semplici operazioni
- Assolutamente **non leggibile** per i non addetti ai lavori
- **Soggetto ad errori** di programmazione
- Necessita di una **profonda conoscenza** dell'hardware

9

il linguaggio assemblativo (assembly)

Per **semplificare il processo** di scrittura dei programmi si

- assegnano alle **locazioni di memoria dei nomi simbolici** (etichette) che dipendono dal problema (ad es. A, B, C, …)
- utilizza, al posto del **codice operativo, un nome mnemonico** che ne ricordi la funzione (ad esempio add, mult, lw, and, or, …)
- utilizza per i **registri dei nomi standard** che dipendono dall'architettura (ad esempio $t0, $t1, $t2, …)



10

esempio

il programma in linguaggio macchina

```
0000 0010 0001
0000 0011 0010
1000 0000 0010 0011
0001 0000 0100
```

in linguaggio assemblativo diventa

```
lw  $t1, A          copia il valore di A nel registro $t1
lw  $t2, B          copia il valore di B nel registro $t2
add $t0, $t1, $t2   somma $t1 + $t2 e poni il risultato in $t0
sw  $t0, C          memorizza il valore di $t0 in C
```

STESSA STRUTTURA MA PIU' COMPRENSIBILE

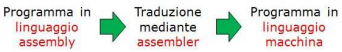
11

il programma assemblatore

Ovviamente un programma scritto in linguaggio assemblativo non e' eseguibile direttamente dal calcolatore

E' necessario un **programma che traduce** il linguaggio assemblativo in linguaggio macchina
(programma assemblatore o Assembler)

- **Fase 1:** traduzione del programma assembly in linguaggio macchina



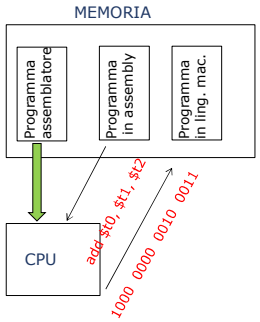
- **Fase 2:** esecuzione del programma in linguaggio macchina

12

il programma assemblatore

- Il programma assemblatore

- Preleva una istruzione in assembly dalla memoria (dati dell'assemblatore)
- Sostituisce mediante una tabella il nome dell'operazione con il corrispondente codice operativo
- Sostituisce al nome delle parole in memoria il corrispondente indirizzo
- Memorizza l'istruzione in linguaggio macchina (risultati dell'assemblatore)



13

linguaggi ad alto livello

Il linguaggio assemblativo risolve **alcuni** problemi del linguaggio macchina
(leggibilita', maggiore facilita' di programmazione)

Rimangono altri problemi
(profonda dipendenza dalla macchina, codici lunghi anche per semplici problemi)

Gia' dagli anni '50 del XX sec. sono nati **linguaggi orientati alle applicazioni**

- FORTRAN (1954): FORmula TRANslator, per usi scientifici
- LISP (1959): LISt Processor, per l'intelligenza artificiale
- COBOL (1960): COmmon Business-Oriented Language, per usi commerciali

14

esempi

Le istruzioni per la somma in linguaggio assembly

```
lw  $t1, A
lw  $t2, B
add $t0, $t1, $t2
sw  $t0, C
```

possono essere riscritte in linguaggio ad alto livello come:

- C = A+B (Fortran)
- c := a+b; (Pascal)
- c = a+b; (linguaggio C)

Meno istruzioni e piu' vicino ad un linguaggio matematico

15

compilatori e interpreti

Analogamente al linguaggio assembly

Anche i linguaggi ad alto livello hanno bisogno di
programmi traduttori
per la traduzione in linguaggio macchina

Due tipi di programmi traduttori

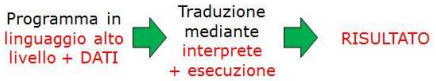
- **Compilatori**
- **Interpreti**

16

l'interprete

L'interprete:

- Preleva un'istruzione dalla memoria
- La traduce in linguaggio macchina
- **esegue immediatamente l'istruzione, prelevando dalla memoria i dati necessari**



In definitiva l'esecuzione del programma avviene in una unica fase, che fornisce direttamente il risultato

17

il compilatore

Il compilatore

- Preleva un'istruzione dalla memoria
- La traduce in linguaggio macchina
- **La conserva in memoria**

In definitiva l'esecuzione del programma avviene in due fasi

- Fase 1: traduzione del programma scritto in linguaggio ad alto livello in linguaggio macchina
- Fase 2: esecuzione del programma scritto in linguaggio macchina

18

differenze tra compilatori e interpreti

Con **l'interprete** l'esecuzione e' costantemente sotto controllo dell'interprete: e' possibile fornire un'istruzione alla volta e controllare subito se e' corretta

(utile in fase di debugging o per ambienti didattici)

Le due fasi distinte richieste dal **compilatore**, rendono possibile la memorizzazione del programma scritto in linguaggio macchina, che e' quindi possibile eseguire numerose volte senza perdere il tempo della traduzione

(maggiore efficienza)

I linguaggi per il **calcolo scientifico** (C e Fortran) sono **linguaggi compilati**

19

in conclusione

Algoritmo: sequenza finita di istruzioni non ambigue per la risoluzione di una classe di problemi

Programma: traduzione di un algoritmo in un linguaggio di programmazione

Fasi per la risoluzione di un problema con il calcolatore:

1. progettazione dell'algoritmo
2. Traduzione in un linguaggio di programmazione
3. Traduzione in linguaggio macchina mediante il compilatore
4. Esecuzione

20

un po' di storia (5)

Fernando Corbato' (1926 - 2019)

- Americano, laureato in fisica al Caltech, comincia a lavorare al centro di calcolo del MIT nel 1956
- Nel 1964 pone le basi e dirige lo sviluppo del primo sistema operativo moderno: il Multics progenitore di Unix e Linux
- Il Multics aveva parecchie caratteristiche presenti ancora oggi nei s.o., tra cui il time sharing e la multiprogrammazione che permettono l'accesso contemporaneo al sistema da parte di piu' utenti
- Riceve numerosi premi tra cui il Turing Award (1990) per i suoi contributi allo sviluppo dei sistemi operativi.

21