UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Architettura degli Elaboratori
Lez. 3 – ASM: Strutture di controllo

Prof. Andrea Sterbini – sterbini@di.uniroma1.it

Argomenti

Argomenti della lezione

-Le strutture di controllo

-Esempi di programmi

-Il simulatore MARS

Scaricate il **simulatore MARS** da

http://courses.missouristate.edu/kenvollmar/mars/index.htm

-è scritto in Java (**gira su qualsiasi OS**)

-è **estensibile** (diversi plug-in per esaminare il comportamento della CPU e della Memoria)

-è un IDE integrato:

- **Editor con evidenziazione della sintassi** assembly ed **autocompletamento** delle istruzioni
- **Help completo** (istruzioni, syscall …)
- **Simulatore** con possibilità di **esecuzione passo-passo** e uso di **breakpoint**
- Permette l'**esecuzione da riga di comando** e la **compilazione di più file**

Assembly MIPS

Direttive principali per l'assemblatore

- **.data** definizione dei dati statici
- **.text** definizione del programma

- **.asciiz** stringa terminata da zero
- **.byte** sequenza di byte
- **.double** sequenza di double
- **.float** sequenza di float
- **.half** sequenza di half words
- **.word** sequenza di words

Codici mnemonici delle **istruzioni**

add, sub, div, beq …

Codifica mnemonica dei **registri**

$a0, $sp, $ra … $f0, $f31

Etichette (per calcolare gli indirizzi relativi)

nome:

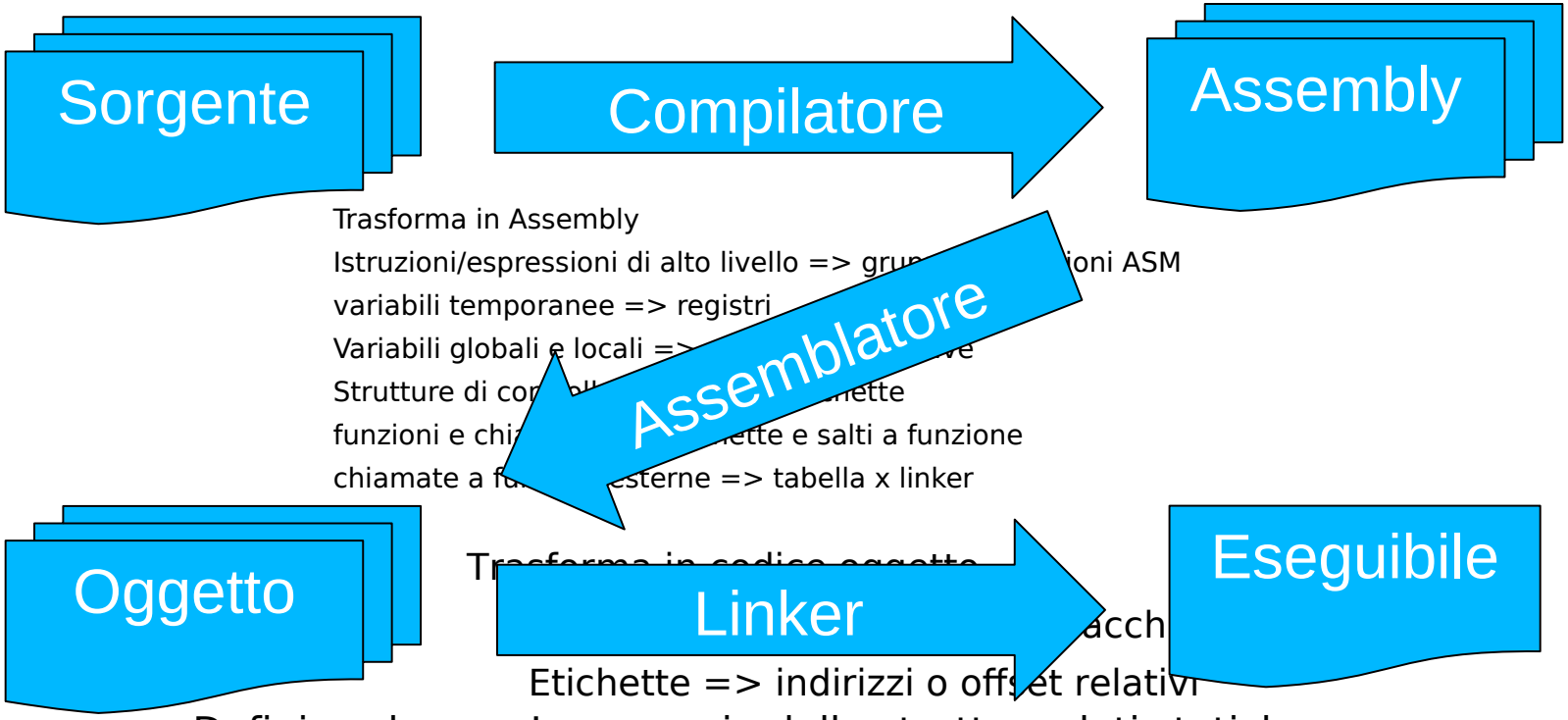
L'assemblatore converte

- dal testo del programma in assembly
- al programma in codice macchina

- Dalle etichette calcola gli indirizzi
  - Dei salti relativi
  - Delle strutture dati in memoria (offset)
  - Dei salti assoluti

NOTA: le **strutture di controllo del flusso** del programma vanno realizzate «a mano» usando i **salti condizionati e le etichette**

Compilatore / Assemblatore



Definisce la pos. In memoria delle strutture dati statiche e del codice

«Collega» i riferimenti a

- chiamate di funzioni esterne => salti non relativi
- strutture dati esterne => indirizzamenti non relativi

Struttura di un programma ASM

.data

definizione dei dati allocati staticamer
partono da 0x 1000 0000
etichetta: .word valori

.text

istruzioni del programma
partono da 0x 0040 0000
oppure dalla etichetta **main**

Salti e condizioni logiche

Condizioni logiche

seq $t0, $t1, $t2 (Set Equal To) $t0 = 1 se $t1==$t2, altrimenti 0

slt $t0, $t1, $t2 (Set Less Than) $t0 = 1 se $t1 < $t2, altrimenti 0

sgt $t0, $t1, $t2 (Set Greater Than) $t0 = 1 se $t1 > $t2, altrimenti 0

…

Operazioni logiche

and $t0, $t1, $t2 (AND bit a bit) $t0 = $t1 && $t2

or $t0, $t1, $t2 (OR bit a bit) $t0 = $t1 || $t2

not $t0, $t1 (NOT bit a bit) $t0 = ! $t1

Salti condizionati

beq $t0, $t1, label (Branch if EQual) salta se $t0 == $t1

ble $t0, $t1, label (Branch if Less or Equal) salta se $t0 <= $t1

blt $t0, $t1, label (Branch if Less Than) salta se $t0 < $t1

…

Salto non condizionato

j label (Jump) salta alla destinazione

IF THEN ELSE

UNITELMA SAPIENZA — SAPIENZA UNIVERSITÀ DI ROMA, DIPARTIMENTO DI INFORMATICA

Esempio C

```
if ( X > 0 ) {

    // codice da eseguire se il test è vero


} else {
    // codice da eseguire se il test è falso

}
    // codice seguente
```

Esempio Assembly

```
.text
    # uso il registro $t0 per la var. X
    blez $t0, else      # test X <= 0

    # codice da eseguire se il test è vero

    j endIF             # esco dall'IF
else:
    # codice da eseguire se il test è falso

endIf:
    # codice seguente
```

NOTA: il test inserito è l'opposto dell'originale

Ciclo DO WHILE

Esempio C

```
do {

    // codice da ripetere se x != 0
    // il corpo del ciclo DEVE
aggiornare x

} while (x != 0);

    // codice seguente
```

Esempio Assembly

```
.text
    # uso il registro $t0 per l'indice x
do:

    # codice da ripetere

    bnez $t0, do  # test x != 0

    # codice seguente
```

NOTA: il test inserito è uguale all'originale

Ciclo WHILE DO

Esempio C

```
while (x != 0) {

    // codice da ripetere se x != 0
    // il corpo del ciclo DEVE aggiornare x
}

    // codice seguente
```

Esempio Assembly

```
.text
    # uso il registro $t0 per l'indice x
while:
    beqz $t0, endWhile      # test x == 0

    # codice da ripetere

    j while                 # loop
endWhile:
    # codice seguente
```

NOTA: il test inserito è l'opposto dell'originale

Ciclo FOR

Esempio C

```
for (i=0 ; i<N ; i++)
{
      // codice da ripetere

}

      // codice seguente
```

Esempio Assembly

```
.text
      # uso il registro $t0 per l'indice i
      # uso il registro $t1 per il limite N
      xor $t0, $t0, $t0          # azzero i
      li $t1, N                  # limite del ciclo
cicloFor:
      ble $t0, $t1, endFor       # test i>=N
      # codice da ripetere
      addi $t0, $t0, 1           # incremento di i
      j cicloFor                 # loop
endFor:
      # codice seguente
```

NOTA: il test inserito è l'opposto dell'originale

SWITCH CASE

Esempio C

```
switch (A) {
case 0:   // codice del caso 0
      break;
case 1:   // codice del caso 1
      break;
// altri casi
case N:   // codice del caso 3
      break;
}
// codice seguente
```

Esempio Assembly

```
.text
      sll $t0, $t0, 2       # A*4
      lw $t1, dest($t0)         # carico indirizzo
      jr $t1                    # salto a registro
caso0:   # codice del caso 0
      j endSwitch
caso1:   # codice del caso 1
      j endSwitch
# altri casi
casoN:   # codice del caso N
      j endSwitch
endSwitch:
      # codice seguente
.data
dest:     .word caso0, caso1, …., casoN
```

Es.: trova il max di un vettore

Esempio C

```
// definizione dei dati
int vettore[6] = { 11, 35, 2, 17, 29, 95 };
int N = 6;

int max = vettore[0];
// scandisco il vettore
for (i=1 ; i<N ; i++) {

      int el = vettore[i];// el. corrente
      if (elemento > max)
      max = elemento;


}
```

Esempio Assembly

```
.data
vettore:    .word 11, 35, 2, 17, 29, 95
N:    .word 6
.text
      lw    $t0, vettore($zero)    # max ⇔ $t0
      lw    $t1, N     # N ⇔ $t1
      li    $t2, 1     # i = 1
for:  bge   $t2, $t1, endFor
      sll   $t3, $t2, 2          # i*4
      lw    $t4, vettore($t3) # el. = vettore[i]
      ble   $t4, $t0, else       # if (el >= max)
      move      $t0, $t4         # max = el.
else:
      addi$t2, $t2, 1          # i++
      j for
```

syscall

Richieste al sistema operativo

Input:

- **$v0**: operazione **richiesta**
- **$a0..$a2,$f0**: eventuali **parametri**

Output:

- **$v0, $f0**: eventuale

Syscall ($v0)	Descrizione	Argomenti ($a0 …)	Risultato ($v0 …)
1	Stampa Intero	Intero	
4	Stampa Stringa	String Address	
5	Leggi Intero		Intero
8	Leggi Stringa	$a0 = buffer address $a1 = num chars.	
10	Fine programma		

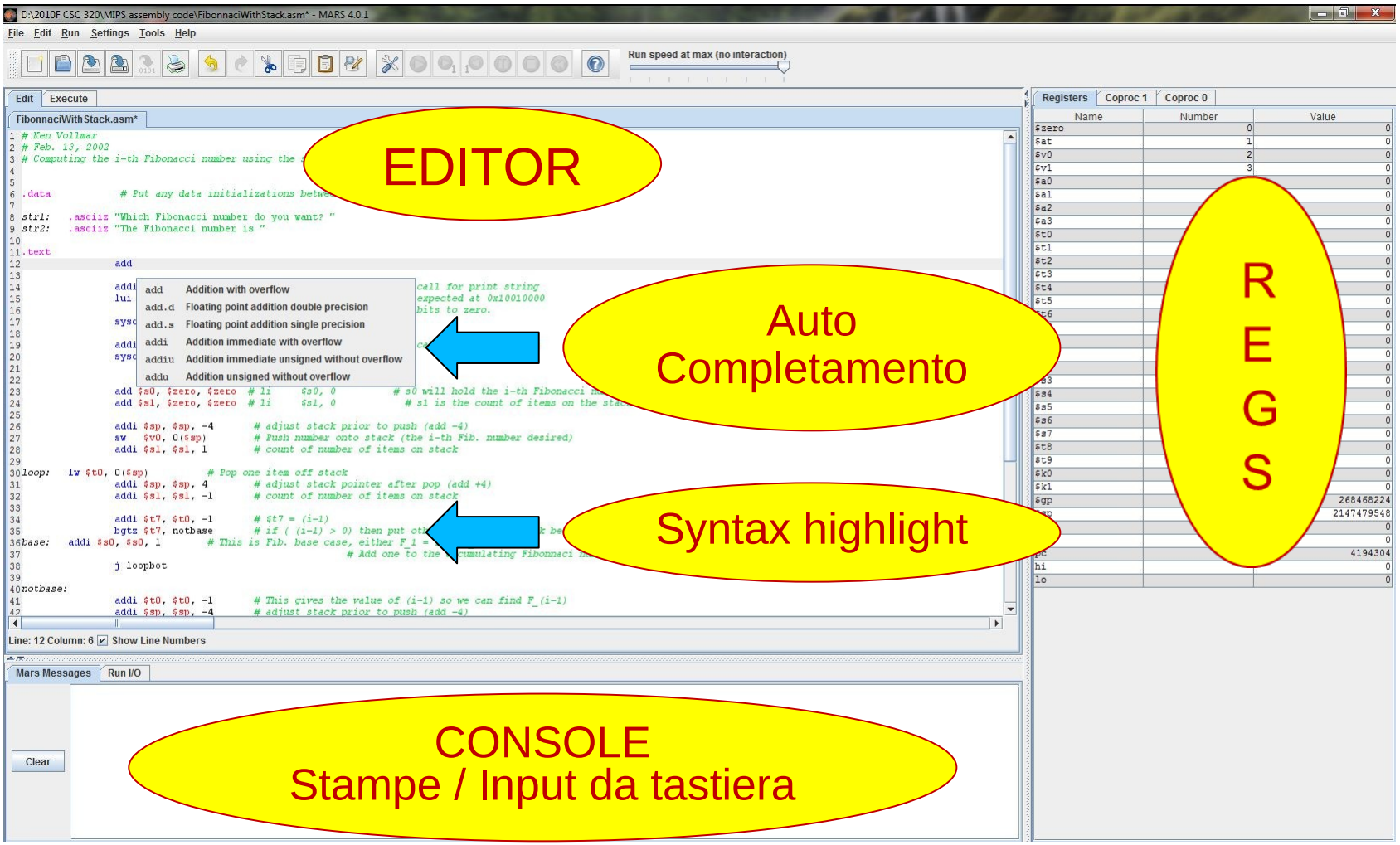
Esempio completo

Esempio C

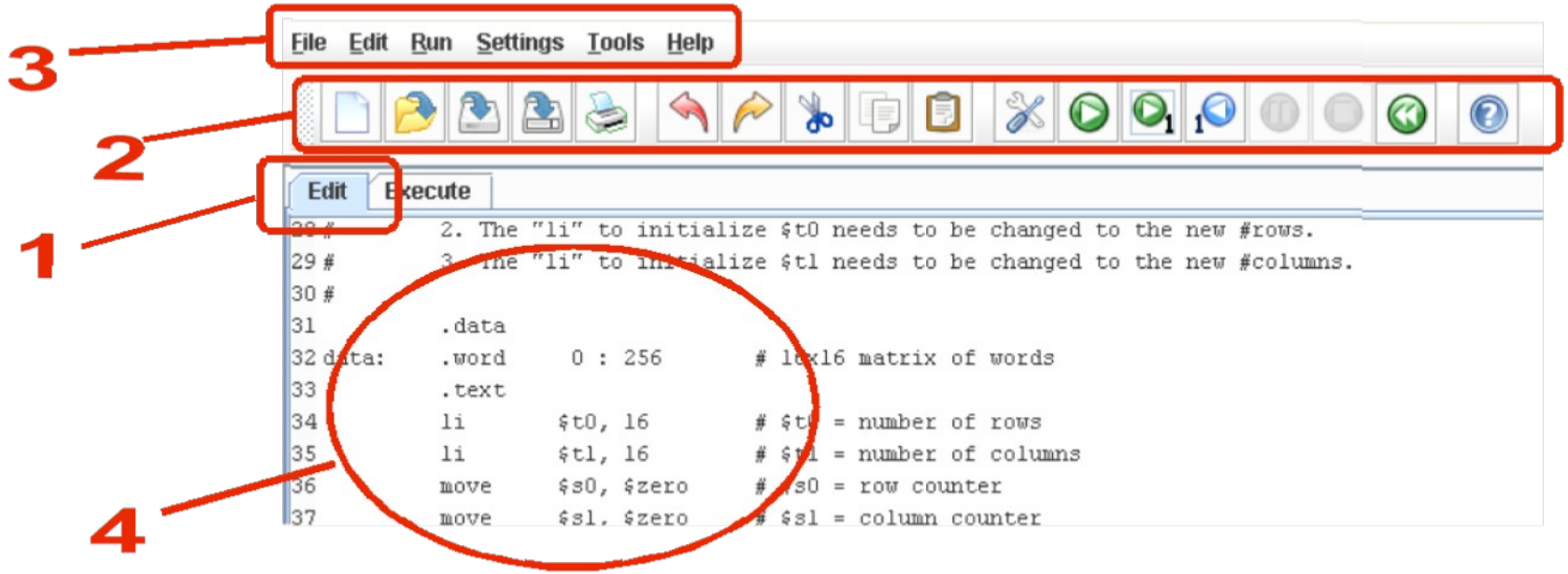
```
// lettura di una serie
// di interi positivi
// terminata da 0
// e stampa del massimo
int main() {
      int max = 0;
      int dato;
      do {
            scanf(«%d», &dato);
            if ( max == 0
                  || dato > max)
                  max = dato;
      } while (dato != 0);
      printf(«%d» , max);
}
```

```
.text     # Codice assembly
main:  move   $t0, $zero   # $t0 ⇔ max
do:    li     $v0, 5   # 5 ⇔ read int
       syscall  # $v0 ⇔ dato
       seq $t1, $t0, $zero  # max == 0
       sgt $t2, $v0, $t0    # dato > max
       or   $t1, $t1, $t2     # ||
       beqz     $t1, endif    # if false endif
       move     $t0, $v0      # max = dato
endif: bnez     $v0, do # se dato != 0
       li     $v0, 1   # 1 ⇔ print int
       move     $a0, $t0      # max
       syscall
       li     $v0, 10 # 10 ⇔ exit
       syscall
```

Il simulatore MARS



MARS: la toolbar



1 Interfaccia a tab (uno per ciascun file aperto, più il simulatore)

2 Toolbar (Compilazione, Esecuzione, Esecuzione passo-passo in avanti e indietro, Reset)

3 Menu

4 Finestra dell'editor