Università degli Studi di Cassino e del Lazio Meridionale

Corso di Calcolatori Elettronici

Costrutti e strutture dati in linguaggio Assembly

Anno Accademico 2015/2016

Francesco Tortorella

Realizzazione delle strutture di controllo in Assembly

In Assembly non esistono le strutture potenti offerte da un HLL, ma è possibile renderle con le istruzioni di controllo Assembly

Strutture di controllo in linguaggio HL	Istruzioni in linguaggio Assembly
• **if-then-else**	• **slt**
• **while**	• **beq,bne**
• **do-while**	• **bgezal**
• **for**	• **j**

Nella realizzazione Assembly delle strutture di controllo sarà necessario gestire esplicitamente molti aspetti che nell'HLL sono risolti in maniera implicita ed automatica.

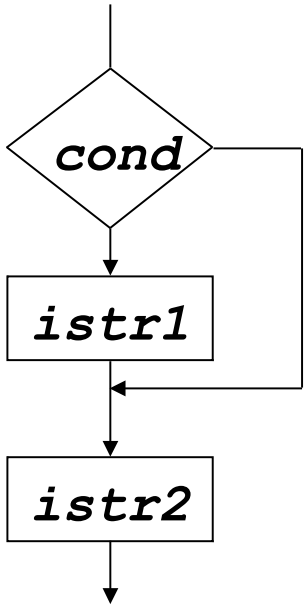
Realizzazione di costrutti di selezione (if-then)

Con le istruzioni di confronto e le istruzioni per il controllo di flusso è possibile realizzare costrutti di selezione del tipo di quelli usati nei linguaggi ad alto livello:

```
if (cond)
   istr1;
istr2;
```

Esempio:

```
if (a>max)
   max=a;
```




```
        if (a<=max)goto next
then    max=a;
next    ...
```

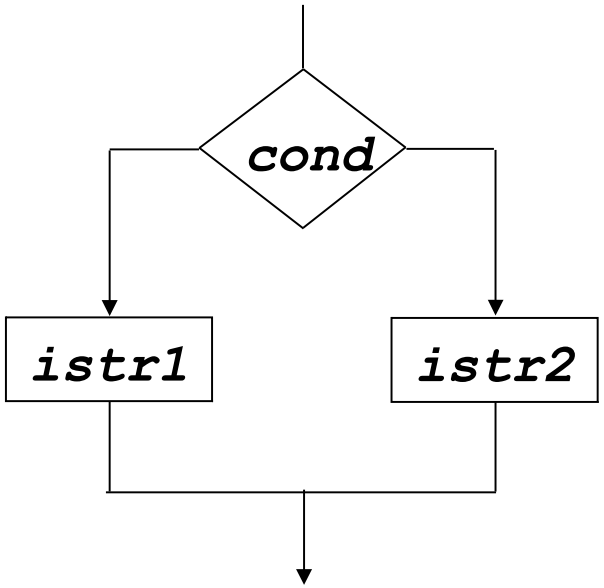
```
        lw    $t0,a
        lw    $t1,max
        ble   $t0,$t1,next
then    sw    $t0,max
next    ...
```

Realizzazione di costrutti di selezione (if-then-else)

Esempio:

```
if (cond)
   istr1;
else
   istr2;
```

```
if (a>b)
   max=a;
else
   max=b;
```




```
        if (a<=b)goto else
then    max=a;
        goto next
else    max=b;
next    …
```

```
        lw    $t0,a
        lw    $t1,b
        ble   $t0,$t1,else
then    sw    $t0,max
        j     next
else    sw    $t1,max
next    ...
```

Costrutti di selezione

- La condizione per il confronto è opposta alla relazione d'ordine nell'`if` perché il salto si effettua verso il blocco sotto l'`else`
- L'uso di `ble` implica che `a,b` e `max` contengano numeri signed. Nel caso fossero stati numeri unsigned si sarebbe dovuto usare `bleu`.

…e nel caso di relazioni più complesse ?

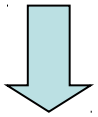
```
if (a>=min && a<=max)
    {Blocco 1}
else
    {Blocco 2}
```

```
        if (a<min) goto else
        if (a>max) goto else
then    {Blocco 1}
        goto next
else    {Blocco 2}
next    …
```

Strutture di controllo: While

Il ciclo while può essere realizzato tramite l'uso contemporaneo di salti condizionati e salti incondizionati

```
  while (a>b)
    {Blocco}
```

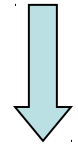


```
while if (a<=b)goto next
      {Blocco}
      goto while
next  …
```

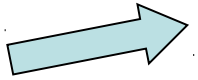
```
      lw   $t0,a
      lw   $t1,b
while ble  $t0,$t1,next
           #{Blocco}
      j    while
next  ...
```

Strutture di controllo: For

```
  for(i=0;i<10;i++)
    {Blocco}
```



```
      ifin=10
      step=1
      i=0
for
      {Blocco}
      i=i+step
      if(i<ifin) goto for
```



```
        li $t1,10     # ifin=10
        li $t2,1      # step=1
        li $t0,0      # i=0
for:
#
#       {Blocco}
#
        add $t0,$t0,$t2    # i=i+step
        blt $t0,$t1,for
```

Strutture di controllo: For

Nel caso in cui nel ciclo non sia necessario accedere alla variabile di conteggio, ma occorra solo assicurare il numero di iterazioni, sono possibili altre soluzioni.

```
        i=9
for
        {Blocco}
        i=i-1
        if(i>=0) goto for
```

```
        li $t2,9
for
#
#       {Blocco}
#
        addi $t2,$t2,-1
        bgez $t2,for
```

Realizzazione di strutture dati in Assembly

Gli unici tipi di dato in Assembly sono **BYTE**, **WORD**, **HALFWORD**.

Non esistono dati strutturati.

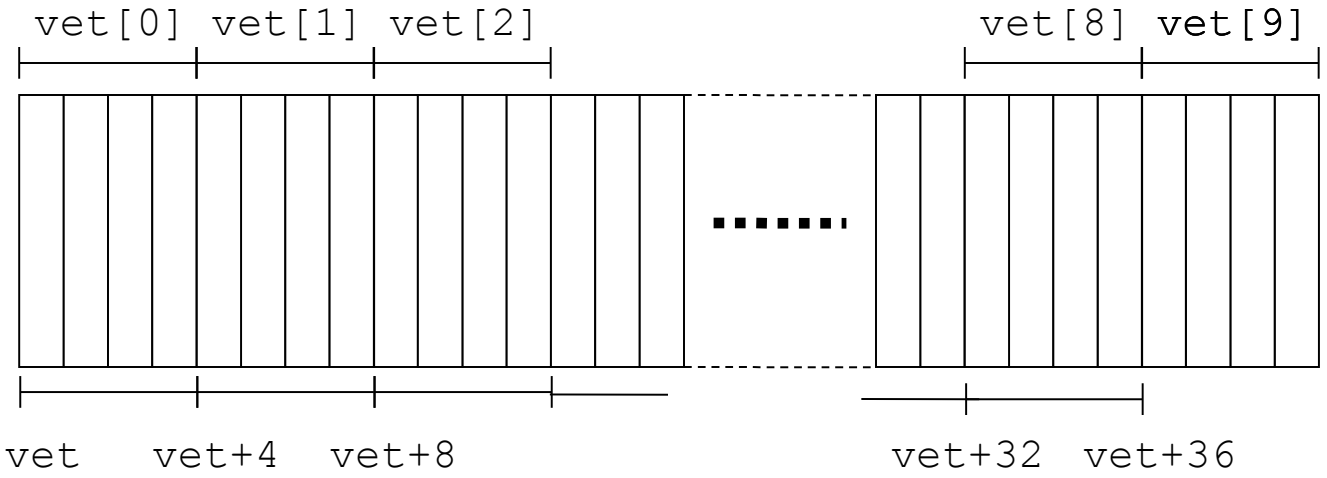
E' necessario gestire esplicitamente tutti gli aspetti relativi alla struttura dati: definizione ed accesso.

HLL	***Assembly***
Nome	Indirizzo
Tipo	Dimensione (# bytes)
Numero di elementi	Numero di bytes allocati

Strutture dati: Array

Definizione

HLL	*Assembly*
int vet[10];	vet **.space** 40



Strutture dati: Array

Accesso

Bisogna costruire la relazione tra l'indice dell'array (compreso tra 0 e N-1) e la locazione di memoria corrispondente.

HLL	***Assembly***
	($s3↔x $s0↔i)
vet[i]=x;	move $t1,$s0 # calcola l'indirizzo
	sll $t1,$t1,2
	sw $s3,vet($t1)# vet[i]=x

Strutture dati: Array

Accesso

Per un accesso sequenziale agli elementi del vettore, la gestione dell'indice potrebbe essere realizzata in modo diverso, utilizzando un puntatore .

HLL	***Assembly*** ($s3↔x $s0↔i)	
`p=vet;`	`la $t1,vet`	`#p=vet`
`x=*p;`	`lw $s3,($t1)`	`# x=p`
…	…	
`p++;`	`addiu $t1,$t1,4`	`#(p++)`

Esempio

```
int vet[10];
int N=10;

for (i=0;i<N,i++)
  vet[i]=0
```

2a soluzione

```
# Inizializzazione
        li    $s0,9
        la    $t1,vet
        # vet[i]=0
for:    sw    $zero,($t1)
        addiu $t1,$t1,4
        addi  $s0,$s0,-1
        bgez  $s0,for
```

1a soluzione

```
# Inizializzazioni
        li   $s1,10    # imax=10
        li   $s2,1     # step=1
        li   $s0,0     # i=0

for:    add $t1,$s0,0
# calcola l'indirizzo di vet[i]
        sll $t1,$t1,2
        # vet[i]=0
        sw  $zero,vet($t1)
        # i=i+step
        add $s0,$s0,$s2
        blt $s0,$s1,for
```