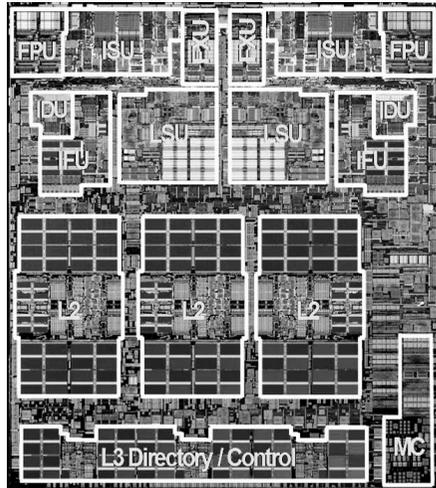




Università degli Studi di Cassino



*Corso di
Calcolatori Elettronici I*

*Costrutti di controllo
Strutture dati*

Anno Accademico 2007/2008

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

- **if-then-else**
- **while**
- **do-while**
- **for**

Istruzioni in linguaggio Assembly

- **slt**
- **beq,bne**
- **bgezal**
- **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)

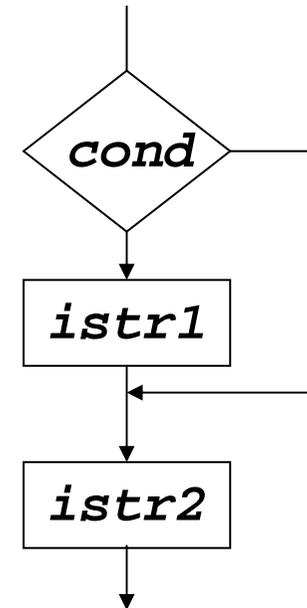
Con le istruzioni di confronto e le istruzioni per il controllo di flusso è possibile realizzare costrutti di selezione del tipo di quelli usati nei linguaggi ad alto livello:

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

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

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

`lw $s0,a`
`lw $s1,max`
`slt $at,$s1,$s0`
`beq $at,$0,next`
`then sw $s0,max`
`next ...`

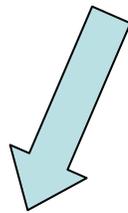


Realizzazione di costrutti di selezione (if-then-else)

Esempio:

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

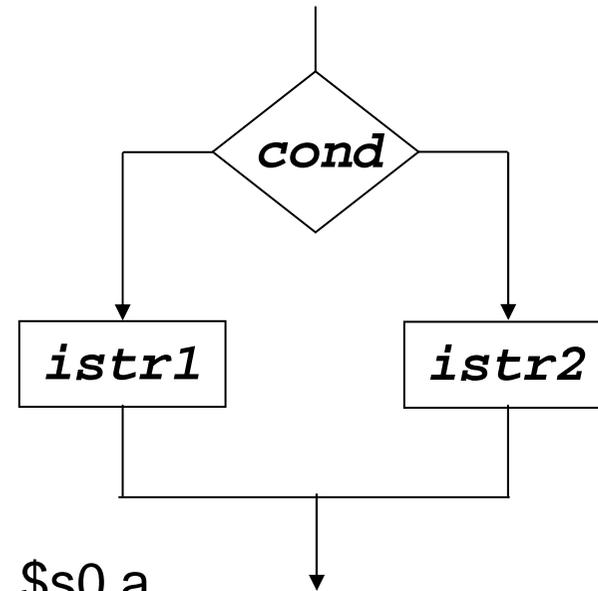
```
if (a>b)
  max=a;
else
  max=b;
```



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



```
lw $s0,a
lw $s1,b
slt $at,$s1,$s0
beq $1,$0,else
then sw $s0,max
j next
else sw $s1,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 `slt` implica che `a`, `b` e `max` contengano numeri signed. Nel caso fossero stati numeri unsigned si sarebbe dovuto usare `sltu`.

...e nel caso di relazioni più complesse ?

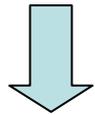
```
if (a>=min && a<=max)
    {Blocco 1}
else
    {Blocco 2}

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}
```



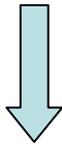
```
while if (a<=b)goto next  
  {Blocco}  
  goto while  
next ...
```



```
lw $s0,a  
lw $s1,b  
while slt $at,$s1,$s0  
beq $at,$0,next  
#{Blocco}  
j while  
next ...
```

Strutture di controllo: For

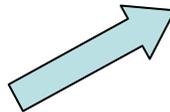
```
for(i=0;i<10;i++)  
  {Blocco}
```



```
ifin=9  
i=0
```

for

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



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

for:

```
#  
#  
#
```

```
{Blocco}
```

```
add $s0,$s0,$s2    # i=i+step  
slt $s3,$s0,$s1  
bne $s3,$0,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.

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



```
for  
#  
#      {Blocco}  
#  
addi $s2,$s2,-1  
bgez $s2,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.

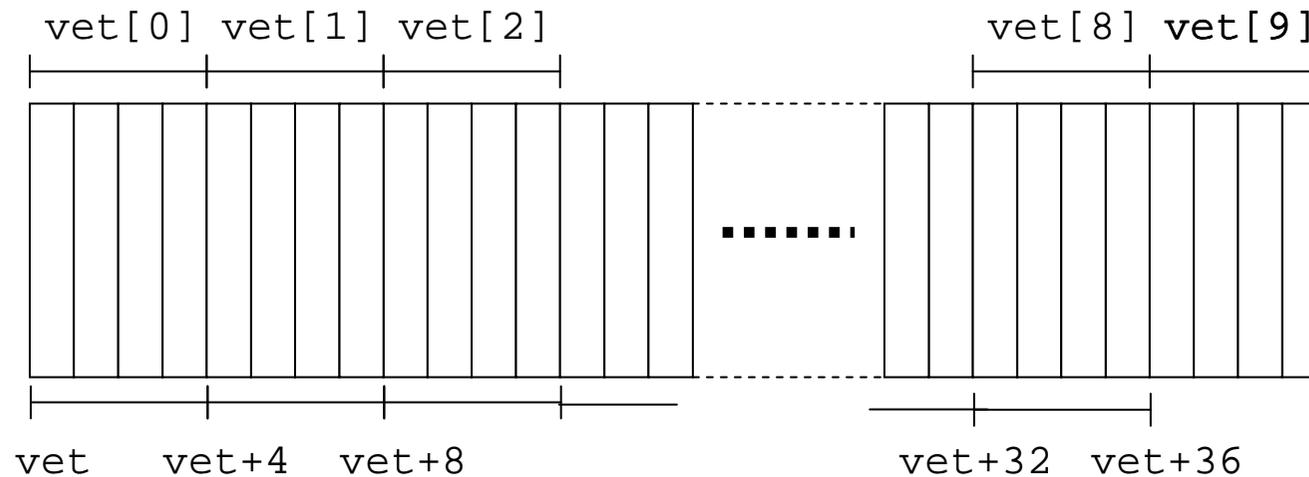
<i>HLL</i>	<i>Assembly</i>
Nome	Indirizzo
Tipo	Dimensione (# bytes)
Numero di elementi	Numero di bytes allocati

Strutture dati: Array

Definizione

HLL
`int vet[10];`

Assembly
`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

```
vet[i]=x;
```

Assembly

```
($s3↔x $s0↔i)
```

```
add $t1,$s0,0 # 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 .

<i>HLL</i>	<i>Assembly</i>
	(\$s3↔x \$s0↔i)
p=veti	la \$t1,vet #p=vet
x=*p;	loop:
	...
p++;	sw \$s3,0(\$t1) # x=p
:	addiu \$t1,\$t1,4 # \$t1<-[\$t1]+4
	# (p++)
	...
	endloop:

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    $zero,0($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
    slt $s3,$s0,$s1
    bne $s3,$0,for
```