CC311 Computer Architecture

Lecture 9
MIPS

Review Example

```
#z=(a*b)+(c/d)-(e+f*g);
        la    $t0,a
        lw    $s0,0($t0)
        la    $t1,b
        lw    $s1,0($t1)
        mul   $t2,$s0,$s1
        la    $t0,c
        lw    $s0,0($t0)
        la    $t1,d
        lw    $s1,0($t1)
        div   $s0,$s1
        mflo  $t1
```

```
        la    $t0,e
        lw    $s0,0($t0)
        la    $t0,f
        lw    $s1,0($t0)
        la    $t0,g
        lw    $s2,0($t0)
        mul   $t3,$s1,$s2
        add   $t3,$s0,$t3
        add   $t2,$t2,$t1
        sub   $t2,$t2,$t3
        la    $t0,z
        sw    $t2,0($t0)
```

Review Example

```
#z[i+4]=(a*b)- (e[i+2] *g);

   Consider a, z in memory as byte, e in memory as half
```

Jumps and conditional branches

Branching control structures

- If-then-else and if-then statements

Looping control structures

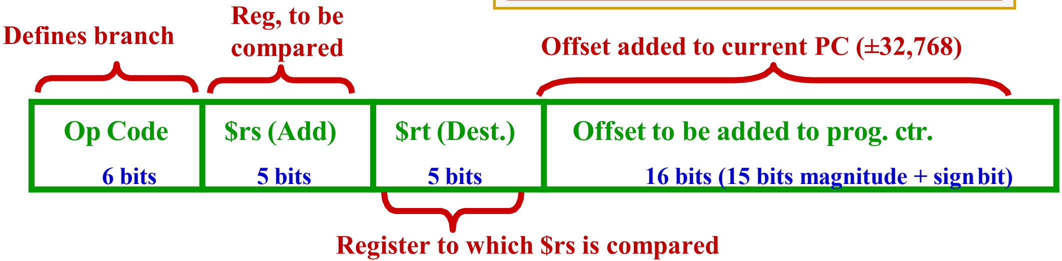
- Do-while, while, and for loops
- Break and continue, indefinite loops

Conditional branching

```
#Basicinstructions
beq    $t1,$t2, label          #if($t1==$t2)goto label
bne    $t1,$t2, label          #if($t1!=$t2)goto label

bgez   $t1, label              #if($t1>=0)goto label
bgtz   $t1, label              #if($t1>0)goto label
blez   $t1, label              #if($t1<=0)goto label
bltz   $t1, label              #if($t1<0)goto  label

#Macroinstructions
beqz   $t1, label              #if($t1==0)goto label
bnez   $t1, label              #if($t1!=0)goto label

beq    $t1, 123, label         #if($t1==123)goto label
bne    $t1, 123, label         #if($t1!=123)goto label

bge    $t1,$t2, label          #if($t1>=$t2)goto label
bgt    $t1,$t2, label          #if($t1>$t2)goto label
bge    $t1, 123, label         #if($t1>=123)goto label
bgt    $t1, 123, label         #if($t1>123)goto label
```

and similarly for `ble` and `blt`

Branch Instructions



Defines branch

Reg, to be compared

Offset added to current PC (±32,768)

Op Code	$rs (Add)	$rt (Dest.)	Offset to be added to prog. ctr.
6 bits	5 bits	5 bits	16 bits (15 bits magnitude + sign bit)

Register to which $rs is compared

- Branch instructions is the I-instruction type .
- In this case, $rs is the register to be compared with something.
- $rt contains the comparison standard. If an immediate (real number) is used, $rt=$at (immediate → $at).
- The op codes for branch instruction are 01 and 04-07.

Review

```
if ((a>b)&&(c==d)) e=0; else e=f;
```

```
la $t0, a
la $t1, b
lw $a0,0($t0)
 lw $a1,0($t1)
la $t0, c
la $t1, d
lw $v0,0($t0)
 lw $v1,0($t1)
```

Review

```
bgt $a0,$a1,cond
j else
cond: beq $v0,$v1,then
else:   la $t0, e
        la $t1, f
        lw $a1,0($t1)
        sw $a1,0($t0)
        j end
then:   la $t0, e
        sw $0,0($t0)
 end:    li $v0,10
         syscall
```

Reading strings into memory

Jumps and conditional branches

Branching control structures

- If-then-else and if-then statements

Looping control structures

- Do-while, while, and for loops
- Break and continue, indefinite loops

Arrays

- For-each loop
- Switch statement

Looping

High level language

```
Do
      ...
      ...
      ...
      ...
      ...
      ...
      ...
      ...
While(cond)
```

Assembly language

```
Dobody:
                ...
                ...   do block
                ...
```

translation of condition, terminating with the label of do block (Dobody)

```
                ...
                ...   Rest of
                ...   program
```

Do-while loop

Example

```
#do {

#m = m - 3
#}while( m < n  * 2)
#Registermappings:m:$t0,n:$t1

          Li     $s0, 2

 loop:    addi   $t0,$t0, -3      #(loop)m = m-3

          mul    $t2,$t1, $s0      #tmp=n   *2

          blt    $t0,$t2, loop  #if(m < tmp)goto loop
```

Optimization: Extract loop invariants

```
         mul    $t2,$t1, $s0        #tmp = n*2

loop:    addi   $t0,$t0, -3      #(loop)m = m-3

         blt    $t0,$t2, loop   #if(m < tmp)goto loop
```

Looping

High level language

```
while (cond)
...
...
...
...
...
...
...
//end while
```

Assembly language

```
whilecond:
```

translation of condition, terminating with the label of while block (**whilebody)**

```
                    j endwhile
whilebody:
                          ...
                          ...      } while block
                          ...
                    j whilecond
endwhile:
                          ...
                          ...      } Rest of program
                          ...
```

While loop

```
#while(m <= z+20)

{m++}

 #n = n%m

#Registers:m:$t0,n:$t1,z:$t2
```

Example

```
        addi $t3,$t2, 20       #tmp = z+20
loop:   ble  $t0,$t3, body    #while(m<=tmp)goto body
        j    end              #goto end
body:   addi $t0,$t0, 1      #(inloop)m++
        j    loop            #endloop,repeat
end:    div  $t1,$t0    #n%m
        mfhi $t1
```

For loop

a for loop syntx

```
for (initialize; condition; update) {
   loop-body
}
```

Equivalent program using while loop

```
initialize
while (condition) {
   loop-body
   update
}
```

Exercise

```
#s = 0

#for(i=0;i<m;i++) {

#s = s+i}

#Registers:m:$t0,i:$t1,s:$t2
```

Example

```
        li   $t2, 0          #s=0
        li   $t1, 0          #i=0
loop:   bge  $t1,$t0, end    #(startloop)ifi>=m goto end
        Add  $t2,$t2,$t1     #sum = sum+i
        addi $t1,$t1, 1      #i=i+1
        j     loop           #(endloop)
end:                         #...
```

Review

```
cin << n;
for (i=3;i<n;i++)
    a[i]=b[i]+10;
        li $v0, 5
        syscall
        li $t0, 4
        li $s0,3              # i in $s0
       la $s2,a               # address of a in $s2
       la $s3,b               # address of b in $s2
```

Review

```
loop:    bge $s0,$v0,end
         add $t0,$s3,$s0     # address of b[i] in $t0
          lb $t1,0($t0)       # b[i]  in  $t1
          addi $s1,$t1,10     # b[i]=b[i]+10
          add $t4,$s2,$s0     # address of a[i] in $t0
          sb $s1,0($t4)       # store into a[i]
          addi $s0,$s0,1      # increment i
          j loop
end:
```

Break and continue

In C-like languages, within loops:

- `break` – exit the loop
- `continue` – skip to the next iteration

Translation of break to assembly

```
j endLabel
```

Translation of continue to assembly

In while loop:

- `j   loopLabel`

In for loop:

- Must execute update first

Exercise

```
#t = 0  #for(i=0;i<m;i++){

#if(i%5>2)continue

#t += i}

#Registers:t=$t0,i=$t1,m=$t2
```

Example

```
         li   $t0, 0         # t=0
         li   $t1, 0         #(init)i=0
loop:   bge   $t1,$t2, end    #if(i>=m) goto end
        rem   $t3,$t1, 5      #tmp=i%5
        bgt   $t3, 2, update  #if(tmp>2)continue
        add   $t0,$t0,$t1     #t += i
update: addi  $t1,$t1, 1      #(update)i++
        j     loop            #(endwhile)
end:                          #...
```

Exercise

- Write a program in MIPS to display n terms of natural numbers and their sum.
- Write a MIPS program to count the number of spaces in a string