CSCI 260 Notes

Ralph Vente Nick Szewczak Anton Goretsky

https://github.com/rvente/CSCI-260-Notes

Contents

2019 02 11

Section 2.7

Branching Instructions

```
beq            $t0, $t1, target      # if $t0 == $t1 then target   // branch on equal
bne            $t0, $t1, target      # if $t0 != $t1 then target   // branch on not equal
j              target                # jump to target              // unconditional jump
slt <dest>, <operand1>, <Operand2>
  # if operand 1 < operand 2, dest <- '1'
  # else "                  ", dest <- '0'
```

Control flow Graph (CFG) useful for charting out conditions so that conversion into assembly is trivial

Note: Professor uses '->' for allocation, '<-' for assignment

Example 1

(1) Example: Implement the following `c` code.

```
if (x == y)
  z = x - y;
else
  z = x + y;
z = 2*z;
```

```
# allocate: x -> $s0; y ->$s1; z -> $s2
        beq $s0, $s1, LThen   # if x = y goto LThen
        add $s2, $s0, $s1     # z <- x + y
        j LendiF              # finish iF-then-else
LThen:  sub $s2, $s0, $s1     # z <- x - y
LEndiF: add $s2, $s2, $s2     # z <- 2z
```

`j LendiF` is needed because we must skip over the next line, LThen. If the jump wasn't present, it would (incorrectly) execute the next line of code.

After the LThen line is executed, control flow goes on to the next line, executing the common condition.

Example 2

(2) Execute the following `c` code.

```
//ex:
if x < y
  x = *p;
else
  x = y;
```

Implementation of program (2).

```
# allocate: x -> $s4, y -> $s3, y -> $s2
      slt $t6, $s4, $s3   # t6 <- 1 if x<y, else 0
      bne $t6, 0, Then    # goto Then if x<y
      addi $s4, $sr, 0    # x <-y
      j   End             # Goto end == done!
Then: lw $s4, 0($s2)      # x<-*p
End: ...
```

Loops

Example 3

(3) Execute the following `c` code.

```
for (i+0; i<n; i++) {
  x = 2*x;
}
y = x;
```

Implementation of program (3) in MIPS

```
    # allocate i -> $s0, x -> $s1, n -> $s2, y -> $s3
       addi   $s0, $zero, 0                    # $s0 = $zero + 0 //i <-0
       slt    $t3, $s2, $s0       # t3 <-1 if n < i else  t3 <-0
       bne    $t3, $zero, EndLP   # if $t0 != $zero then target
       add    $s1, $s1, $s1       # $s1 = 2x
       addi   $s0, $s0, 1         # $s0 = $t1 + 1 // i++
       j              LP          # jump to LP //itterate
endLP: add    $s3, $s1, $zero     # $s3 = $t1 + $t2

#alloc. i -> $s3, j -> $s4, k -> $s5, save -> $s6

Loop: add $t7, $s3, $s3   # t7 <- 2i
      add $t7, $t7, $t7   # t7 <- 4i
      add $t7, $t7, $s6   # t7 <- addr. of save[:]
      lw  $t7, 0(st7)     # t7 <- element value
      bne $t7, $s5, Exit  # Exit loop if save [i] != K
      add $s3, $s3, $s4   # i <- i + j
      j   Loop            # next iteration
Exit:
```