MONASH University

Information Technology

FIT1008/2085
MIPS – Selection

Prepared by:
Maria Garcia de la Banda
Revised by A. Aleti, D. Albrecht, G. Farr, J. Garcia and P. Abramson

Where are we up to?

- Know the MIPS R2000 architecture and can name the main parts
- Understand the fetch-decode-execute cycle
- Able to use assembler directives
- Can program in assembly using the MIPS instruction set we will use
- Know the basics of the instruction formats
- Know how to translate simple programs with if-then constructs
- Know how to do it faithfully

FLUX: B23U3R

We must translate the condition of the if-then `if x == y`

Assume the value of `x` is already loaded in `$t0` and that of `y` in `$t1`. The translation can be achieved in MIPS as:

A) `beq $t0, $t1, endif`

B) `bne $t0, $t1, endif`

C) `slt $t2, $t0, $t1` **and** `beq $t2, $0, endif`

D) `slt $t2, $t0, $t1` **and** `bneq $t2, $0, endif`

FLUX: B23U3R

What are the following instructions implementing:

```
slt $t0, $t1, $t2
bne $t0, $0, label
```

A) `if $t1 >= $t2:`

B) `if $t1 < $t2:`

C) `if $t1 <= $t2:`

D) `if $t1 > $t2:`

E) None of the above

Learning objectives for this lecture

- To be able to translate more complex selection: `if-then-else`
- To be able to translate iteration: `for` and `while` loops

MONASH University

Translating if-then-elses

Selection: if-then-else

```
# Sane people write
# code like this.

i = 0

read(i)

if i < 0:

    print(-5 * i)

else:

    print(5 * i)
```

```
# if Python had "goto" you
# could write it like this
# (ugh)

i = 0

read(i)

if not i < 0:
    goto else

    print(-5 * i)

    goto endif

else:

    print(5 * i)

endif:
```

if-then-else in MIPS

Service	Code	Arg	Res
Print integer	1	$a0	n/a
Print string	4	$a0	n/a
Read integer	5	n/a	$v0
Read string	8	$a0 $a1	n/a
Allocate memory	9	$a0	$v0
Exit program	10	n/a	n/a

We will again treat it as a global variable

```
i = 0
read(i)
if i < 0:
    print(-5 * i)
else:
    print(5 * i)
```

```
    .data
i:  .word 0
    .text
# Read integer and store it in i
    addi  $v0, $0, 5
    syscall
    sw   $v0, i
# Comparison part:if not i < 0: goto else
    lw   $t0, i          # load i
    slt $t1, $t0, $0     # is i<0?
    beq $t1, $0, else    # if not, go to else

# fall through.
# and print out -5*i

# continued next column ...
```

if-then-else in MIPS

```
i = 0
read(i)
if i < 0:
    print(-5 * i)
else:
    print(5 * i)
```

```
# print -5*i
addi $t1, $0, -5
lw   $t0, i      # load i
mult $t0, $t1    # -5*i
mflo $a0         #$a0 = -5*i
addi $v0, $0, 1 # print int
syscall
j endif          #jump over else
```

```
else:  # Print 5*i
addi $t1, $0, 5
lw   $t0, i      # load i
mult $t0, $t1    # 5*i
mflo $a0         # $a0 = 5*i
addi $v0, $0, 1 # print int
syscall
```

```
endif:  # Join up here to exit
addi $v0, $0, 10
syscall
```

Your turn – what does this do?

```
                .text
1020   main:  addi $t1,$0,5          # $t1 ← 5
1024          slti $t0,$t1,2         # $t1 < 2? No, so: $t0←0
1028          beq  $t0,$0,foo
102c          addi $t1,$t1,1
1030          j    end
1034    foo:  addi $t1,$t1,-1         # $t1 ← 4
1038    end:   …
```

What does it do?

If `$t1 < 2`, it adds `1` to `$t1`, otherwise it subtracts `1` from it

MONASH University

Translating loops

Reminder: Iteration

- **Iteration is the repetition of a section of code**
  - In Python, with `while`, `for`
  - `while` tests condition before loop entry
  - `for` is a shorthand for `while`
- **Achieved by sending control from the end of the loop back to the beginning**
  - Test some condition to prevent infinite loop

Iteration: while

```
# while loops written
# like this ...

n = 0
f = 1

read(n)

# Compute factorial
while n > 0:
    f = f * n
    n = n- 1

print(f)
```

```
# ... could also be written
#  like this if Python had goto

n = 0
f = 1

read(n)

# Exit if condition=false
loop:
    if not n > 0:
        goto endloop
    # Body of loop.
    f = f * n
    n = n - 1
    # Repeat loop.
    goto loop
endloop:
    print(f)
```

Notice again the negation of the condition

while in MIPS

```
n = 0
f = 1
read(n)
while n > 0:
    f = f * n
    n = n- 1
print(f)
```

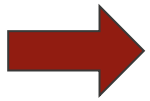
```
        .data
n:      .word 0
f:      .word 1

        .text
        # Read int, store in n
        addi $v0, $0, 5
        syscall
        sw   $v0, n

        # Now comes the loop.
        # Exit loop if not n > 0
loop:   lw   $t0, n          # load n
        slt $t1, $0, $t0     # is n>0?
        beq $t1, $0, endloop #if not,go
        # ... else fall through.

# Continued at right ...
```

```
# ... Continued from left
        # Body of while loop.
        # f = f * n
        lw   $t0, f     # load f
        lw   $t1, n     # load n
        mult $t0, $t1 # f*n
        mflo $t0        # $t0=f*n
        sw   $t0, f     # f = f*n
        # n-- (n = n - 1)
        lw   $t0, n     # load n
        addi $t0, $t0, -1
        sw   $t0, n     # n=n-1

        # End of loop, go back
        j   loop

endloop:# Print integer f
        addi $v0, $0, 1
        lw   $a0, f
        syscall

        addi $v0, $0, 10 # exit
        syscall
```

Iteration: `for`

- **A `for` loop is essentially a simpler version of a `while` loop:**
  - Initialization, condition and increment code all in one place
- **To translate a `for` loop into MIPS, write it as a `while` loop**

```
for i in range(start,stop,step):

    body
```



```
i = start
while i != stop:

    body
    i = i + step
```

Might need to be > or < depending on step

Iteration: for

```
# turn this for loop...

n = 0
read(n)

# Print n 9 times
for counter in
        range( 10,
               1,
               -1 ):

    print(n)
```

```
# into this while loop

n = 0
read(n)

# Print n 9 times
counter = 10
while counter != 1:

    print(n)
    counter -= 1
```

For now, we will treat it as a global

for in MIPS

```
n = 0
counter = 10
read(n)
while counter != 1:
    print(n)
    counter = counter - 1
```

```
    .data

      n: .word 0
counter: .word 10

      .text

      # Read integer, store in n
      addi $v0, $0, 5
      syscall
      sw $v0, n

loop: # For loop: condition.
      # If counter == 1 go endloop
      lw $t0, counter # load counter
      addi $t1, $0, 1
      beq $t0, $t1, endloop

      # Continued at right ...
```

```
# ... Continued from left
# For loop body: print n
addi $v0, $0, 1
lw $a0, n
syscall

# For loop decrement counter.
lw $t0, counter
addi $t0, $t0, -1
sw $t0, counter

# Return to loop start.
j loop

endloop:
      # Exit program
      addi $v0, $0, 10
      syscall
```

Summary

- Now able to translate more complex selection: `if-then-else`
- Now able to translate iteration: `for` and `while` loops

Going further

- **`if-elif-else` statement (`switch`)**
  - Efficiently selecting one of many options
  - Sometimes implemented with jump table (array of target addresses)
- **MIPS jump/branch delay slots**
  - Related to the instruction pipeline
  - Real MIPS CPU executes instruction after a branch or jump because it has already entered the instruction pipeline
  - Ignored in SPIM (not sure about MARS): configured for no delay slots