TUTORIAL 6 MIPS BRANCHES AND JUMP INSTRUCTIONS

Overview

- **You will review/learn the following in this tutorial:**
  - MIPS conditional branch and jump instructions
- **You will practice to write short MIPS programs:**
  - With branches and loops

MIPS syscall

- **A number of system services, mainly for input and output, are available for use by your MIPS program.**
- **syscall is used to request a such service from the kernel.**
  - Step 1 Load the service number in register $v0.
  - Step 2 Load argument values, if any, in $a0, $a1, $a2, or $f12 as specified.
  - Step 3 Issue the SYSCALL instruction.
  - Step 4 Retrieve return values, if any, from result registers as specified.

Available Services

- http://courses.missouristate.edu/kenvollmar/mars/Help/SyscallHelp.html

Service	Code in $v0	Argument(s)	Result(s)
Print Integer	1	$a0 = number to be printed	
Print Float	2	$f12 = number to be printed	
Print Double	3	$f12 = number to be printed	
Print String	4	$a0 = address of string in memory	
Read Integer	5		number returned in $v0
Read Float	6		number returned in $f0
Read Double	7		number returned in $f0
Read String	8	$a0 = address of input buffer in memory $a1 = length of buffer (n)	
Sbrk	9	$a0 = amount	address in $v0
Exit	10		

Example: Sum of Two User-input Integers

```
1  # Add two numbers
2  # $t0: 1st number
3  # $t1: 2nd number
4  # $v0: syscall id and return value
5  # $a0: syscall argument
6
7  .text
8  .globl main
9
10 main:
11
12 # read the 1st nunber from keyboard
13 li $v0, 5 # read int
14 syscall # the int is in $v0
15 move $t0, $v0 # pseudo instruction
16
17 # do similiar for the 2nd nunber
18 li $v0, 5 # read int
19 syscall # the int is in $v0
20 move $t1, $v0 # pseudo instruction
21
22 add $t2, $t0, $t1 # addition
23
24 # print int
25 move $a0, $t2
26 li $v0, 1 # value in $a0 is printed
27 syscall
28
29 # exit
30 li $v0, 10
31 syscall
32
```

- Syscall code
- Syscall input argument(s)
- Syscall return

Example: Sum of Two Random Generated Integers

- **In C++, to generate random integers**
  - Get system time
  - Set system time as seed (only need to set seed for once)
  - Generate random number (for as many times as needed)
  - % operation to put the number in range [0, divisor – 1]
  - Adjust the number with offset
- **Do similar steps in MIPS with the help of syscall**

```
16  #------- Text Segment ----------
17  .text
18  .globl main
19
20  main:
21
22  # get system time
23  li $v0, 30 # syscall 30 time(), returns system time to $a0
24  syscall
25
26  # set rand seed (as system time)
27  move $a1, $a0 # syscall 40 srand(), $a1 holds the seed
28  li $a0, 0
29  li $v0, 40
30  syscall
31
32  li $a1, 20 # assume the range is [1, 20]
33
34  # generate random integer within range
35  li $a0, 0 # syscall 42 rand()%range, $a1 upper bound of range of returned values
36  li $v0, 42
37  syscall # $a0 holds a random int in [0, range-1]
38  addi $t0, $a0, 1 # $t0 holds a random int in [1, range]
39
40  # generate another random integer
41  li $a1, 20 # assume the range is [1, 20]
42  li $a0, 0 # syscall 42 rand()%range, $a1 upper bound of range of returned values
43  li $v0, 42
44  syscall # $a0 holds a random int in [0, range-1]
45  addi $t1, $a0, 1 # $t1 holds a random int in [1, range]
46
47  add $t2, $t0, $t1 # addition
48
```

Conditional and Unconditional Jumps

Branch on Equal	Usage: beq <reg1>, <reg2>, <target>
If the values stored in reg1 and reg1 are equal, jump to target	
Branch on Non-equal	Usage: bne <reg1>, <reg2>, <target>
If the values stored in reg1 and reg1 are not equal, jump to target	
Jump	Usage: j <target>
Jumps to the calculated address	

In-equality Comparison

Set on Less Than	Usage: slt <reg 1>, <reg 2>, <reg 3>
Register reg1 is set to 1 if the value in reg2 is less than the value in reg3; otherwise, register reg1 is set to 0	
Set on Less Than Immediate	Usage: slti <reg 1>, <reg 2>, <immediate>
Register reg1 is set to 1 if the value in reg2 is less than the immediate; otherwise, register reg1 is set to 0	

Warm up exercise

- **Write down the shortest sequence (any one) of MIPS instructions for the following C++ code, assuming variable d is stored in $s0**

```
if (d == 1){
    d = d + 3;
}
```

Solution:

```
addi $t0, $zero, 1
bne $t0, $s0, Exit
addi $s0, $s0, 3
Exit:
```

Example: Absolute Value

```
1   .text
2   .globl main
3
4   # find the absolute value of $s0 output it
5   main:
6           # read the 1st nunber from keyboard
7           li $v0, 5 # read int, pseudo instruction
8           syscall # the int is in $v0
9           move $s0, $v0 # pseudo instruction
10
11          slt $t0, $s0, $zero # is ($s0 < 0)?
12          beq $t0, $zero, print # go directly to print if $s0 >= 0
13  negative:
14          sub $s0, $0, $s0 # 'flip' the negative value
15  print:  add $a0, $s0, $zero # print the absolute value
16          li $v0, 1
17          syscall
18  exit:   li $v0, 10 # exit
19          syscall
```

Example: Add Even Integers

```
1   .text
2   .globl main
3
4   main:
5           li $s0, 100 # add up all even numbers in range [1, $s0], li pseudo instruction
6           li $t0, 1 # $t0, loop iterator
7           li $t1, 0 # $t1, sum
8
9   loop:
10          slt $t2, $s0, $t0 # if (i <= 100), which is equivalent if !(100 < i)
11          bne $t2, $zero, exit
12
13          andi $t3, $t0, 1 # $t3 either 0000...000 (when $t0 is even) or 0000...001 (when $t0 is odd)
14
15          bne $t3, $0, next_iteration # if i is odd, skip the addition
16          add $t1, $t1, $t0 # sum += i if i is even
17
18  next_iteration:
19          addi $t0, $t0, 1 # i++
20          j loop
21
22  exit:
23          li $v0, 10
24          syscall
25
```

Example: Simple for-Loop

```
1  # print given number of stars
2  .data
3  msg:    .asciiz "How many stars to print? \n"
4  star:   .asciiz "*"
5
6  .text
7  .globl main
8
9  main:
10         # print msg
11         la $a0, msg
12         li $v0, 4
13         syscall
14
15         # read user input
16         li $v0, 5
17         syscall # the int is in $v0
18         move $s0, $v0 # $s0 the number of stars to print
19
20         la $a0, star
21         li $t0, 0 # $t0, loop iterator, init as 0
22 loop:
23         beq $t0, $s0, exit # if ($t0 < # stars) continue loop
24         li $v0, 4 # print string with starting addr in $a0
25         syscall
26
27         addi $t0, $t0, 1
28         j loop
29 exit:
30         li $v0, 10
31         syscall
```

Exercise

- **Write down the shortest sequence (any one) of MIPS instructions for the following C++ code, assuming variable c and the base address of int array A are stored in $s0 and $s1 respectively. You can use some registers for storing temporary values.**

```
c = 0;
do {
    c = c + 2;
    A[c - 1] = A[c];
} while (c < 10);
```

Solution:

```
addi $s0, $zero, 0
Loop: addi $s0, $s0, 2
      sll $t0, $s0, 2
      add $t0, $s1, $t0
      lw $t1, 0($t0)
      sw $t1, -4($t0)
      slti $t2, $s0, 10
      bne $t2, $zero, Loop
```

Loop Unfurling:

```
lw $t0, 8($s1)
sw $t0, 4($s1)
lw $t0, 16($s1)
sw $t0, 12($s1)
lw $t0, 24($s1)
sw $t0, 20($s1)
lw $t0, 32($s1)
sw $t0, 28($s1)
lw $t0, 40($s1)
sw $t0, 36($s1)
addi $s0, $zero, 10
```

Example: Max in an Array

```
10  #------- Text Segment ----------
11  # $s0: array starting address
12  # $s1: array size
13  # $s2: max, init as array[0]
14
15  # $t0: loop iterator i
16  # $t1: address of array[i]
17  # $t2: content of array[i]
18
19  # max = array[0]
20  # for (int i = 1; i < size; i++)
21  #       if (array[i] > max)
22  #               max = array[i];
23
24  .text
25  .globl main
26
27  main:
28          la $s0, array           # $s0: array starting address
29          la $s1, size
30          lw $s1, 0($s1)          # $s1: array size
31
32          lw $s2,0($s0)           # $s2: max, init as array[0]
33
34          li $t0,1                # $t0: loop iterator i, init as 1
35  max_loop:
36          beq $t0,$s1,print_array    # if (i<size) continue loop
37
38          sll $t1,$t0,2           # $t1 = 4*i
39          add $t1,$t1,$s0         # $t1: addr of array[i]
40          lw  $t2,0($t1)          # $t2: content of array[i]
41          slt $t3,$s2,$t2         # if ( max < array[i])
42          beq $t3,$zero,max_next_iteration # no, next iteration
43          add $s2,$t2,$zero       # yes, max = array[i]
44
45  max_next_iteration:
46          addi $t0,$t0,1     # i++
47          j   max_loop
```

Example: Nested for-Loop

- Use proper labels
- Maintain a good ‘variable-register’ mapping table

```
19  # $s0: size of RAT
20  # top row (row 0): 1 star
21  # 2nd row  (row 1) : 2 stars
22  # row i: i+1 stars
23
24  # $t0: i, $t1: j
25  # for (int i = 0; i < size; i++) {
26  #    for (int j = 0; j <= i; j++)
27  #        cout << "*";
28  #    cout << endl;
29  # }
30
31  main:
32          la $a0, msg1            # print msg
33          li $v0, 4
34          syscall
35
36          li $v0, 5               # read int
37          syscall                 # the int is in $v0
38          move $s0, $v0           # pseudo instruction
39
40          la $a0, RAT             # print msg
41          li $v0, 4
42          syscall
43
44          li $t0, 0               # i = 0
45  outer_loop:
46          slt $t2,$t0,$s0         # if (i<size) loop
47          beq $t2, 0, exit
48
49          move $t1, $0
50  inner_loop:
51          slt $t2, $t0, $t1       # if (j <= i) is equivalent to if !(i < j)
52          bne $t2, $0, print_enter
53          la $a0, star            # draw a star
54          li $v0, 4
55          syscall
56  increment_j:
57          addi $t1, $t1, 1
58          j inner_loop
59
60  print_enter:
61          la $a0, newLine         # print enter
62          li $v0, 4
63          syscall
64  increment_i:
65          addi $t0, $t0, 1
66          j outer_loop
67
68  exit:
69          li $v0, 10
70          syscall
```

Extra Exercise

- **Write down the shortest sequence (any one) of MIPS instructions for the following C++ code, assuming variable d is stored in $s0. You can use some registers for storing temporary values.**

```
if (d < 4) {
    if (d == 1)
        d = d + 4;
    else ++d;
}
```

Solution:

```
slti $t0, $s0, 4
beq $t0, $zero, Exit
addi $t1, $zero, 1
bne $s0, $t1, Else
addi $s0, $s0, 4
j Exit
Else: addi $s0, $s0, 1
Exit:
```

Alternative Solution:

```
slti $t0, $s0, 4
beq $t0, $zero, Exit
addi $s0, $s0, 1
addi $t1, $zero, 2
bne $s0, $t1, Exit
addi $s0, $s0, 3
Exit:
```

Extra Exercise

- **Write down the shortest sequence (any one) of MIPS instructions for the following C++ code, assuming variable d is stored in $s0. You can use some registers for storing temporary values.**

```
switch (d) {
    case 1: d = d + 4;
            break;
    case 4: d = d * 2;
            break;
    default: d--;
}
```

Solution:

```
addi $t1, $zero, 1
bne $s0, $t1, case4
addi $s0, $s0, 4
j exit
case4: addi $t4, $zero, 4
       bne $s0, $t4, default
       sll $s0, $s0, 1
       j exit
default: addi $s0, $s0, -1
exit:
```

Alternative Solution:

```
addi $s0, $s0, -1
bne $s0, $zero, case4
addi $s0, $zero, 5
j exit
case4: addi $t4, $zero, 3
       bne $s0, $t4, exit
       addi $s0, $zero, 8
exit:
```

Extra Exercise

- **Write down the shortest sequence (any one) of MIPS instructions for the following C++ code, assuming variable c and the base address of int array A are stored in $s0 and $s1 respectively. You can use some registers for storing temporary values.**

```
for (int c = 0; c <= 10; c += 2){
    A[c] = A[c + 3];
}
```

Solution executes 6*7 + 1 = 50 instructions.

Loop Unfurling executes 12 instructions.

Tradeoff between code length and readability vs execution time.

Solution:

```
addi $s0, $zero, 0
loop: sll $t0, $s0, 2
      add $t1, $s1, $t0
      lw $t2, 12($t1)
      sw $t2, 0($t1)
      addi $s0, $s0, 2
      slti $t3, $s0, 11
      beq $t3, $zero, loop
```

Loop Unfurling:

```
lw $t0, 12($s1)
sw $t0, 0($s1)
lw $t0, 20($s1)
sw $t0, 8($s1)
lw $t0, 28($s1)
sw $t0, 16($s1)
lw $t0, 36($s1)
sw $t0, 24($s1)
lw $t0, 44($s1)
sw $t0, 32($s1)
lw $t0, 52($s1)
sw $t0, 40($s1)
```