TUTORIAL 6 MIPS BRANCHES AND JUMP INSTRUCTIONS

Overview

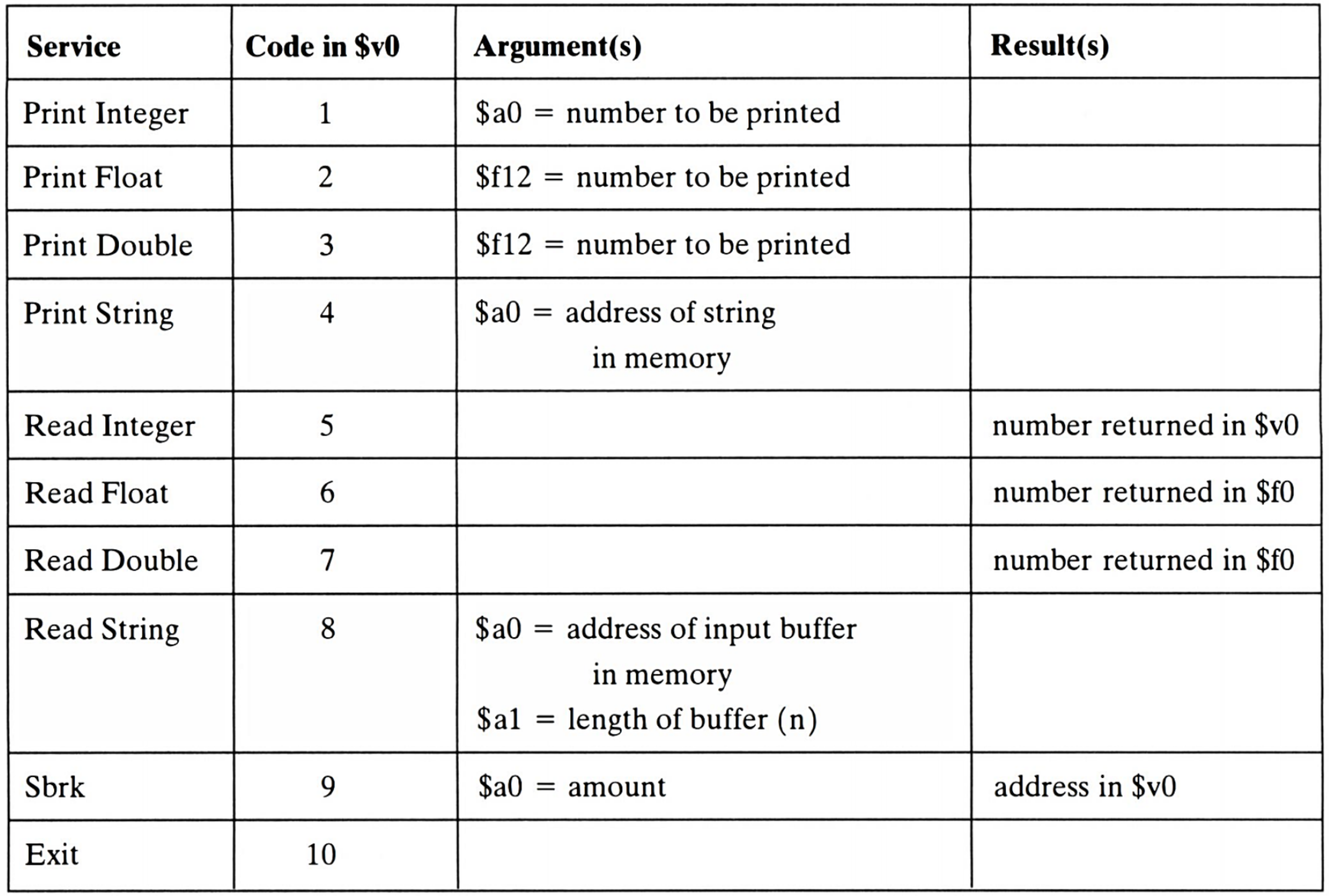
- **You will review/learn the following in this tutorial:**
  - MIPS conditional branch and jump instructions
- **You will practice to write short MIPS programs:**
  - With branches and loops

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.

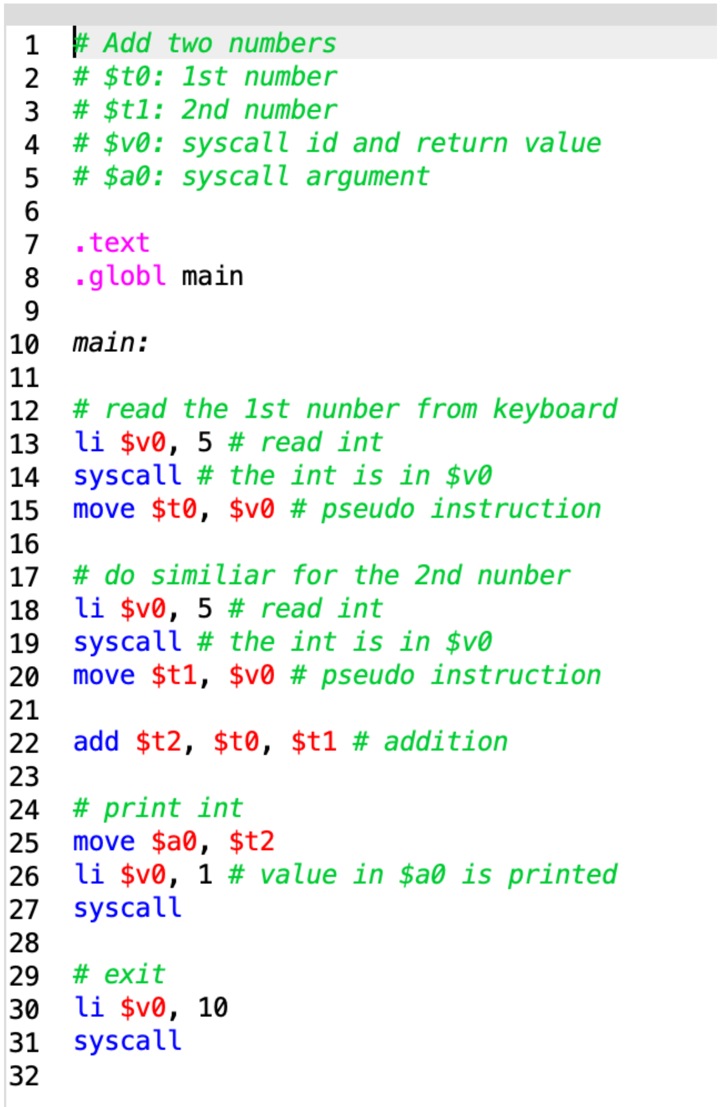
Available Services

- 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

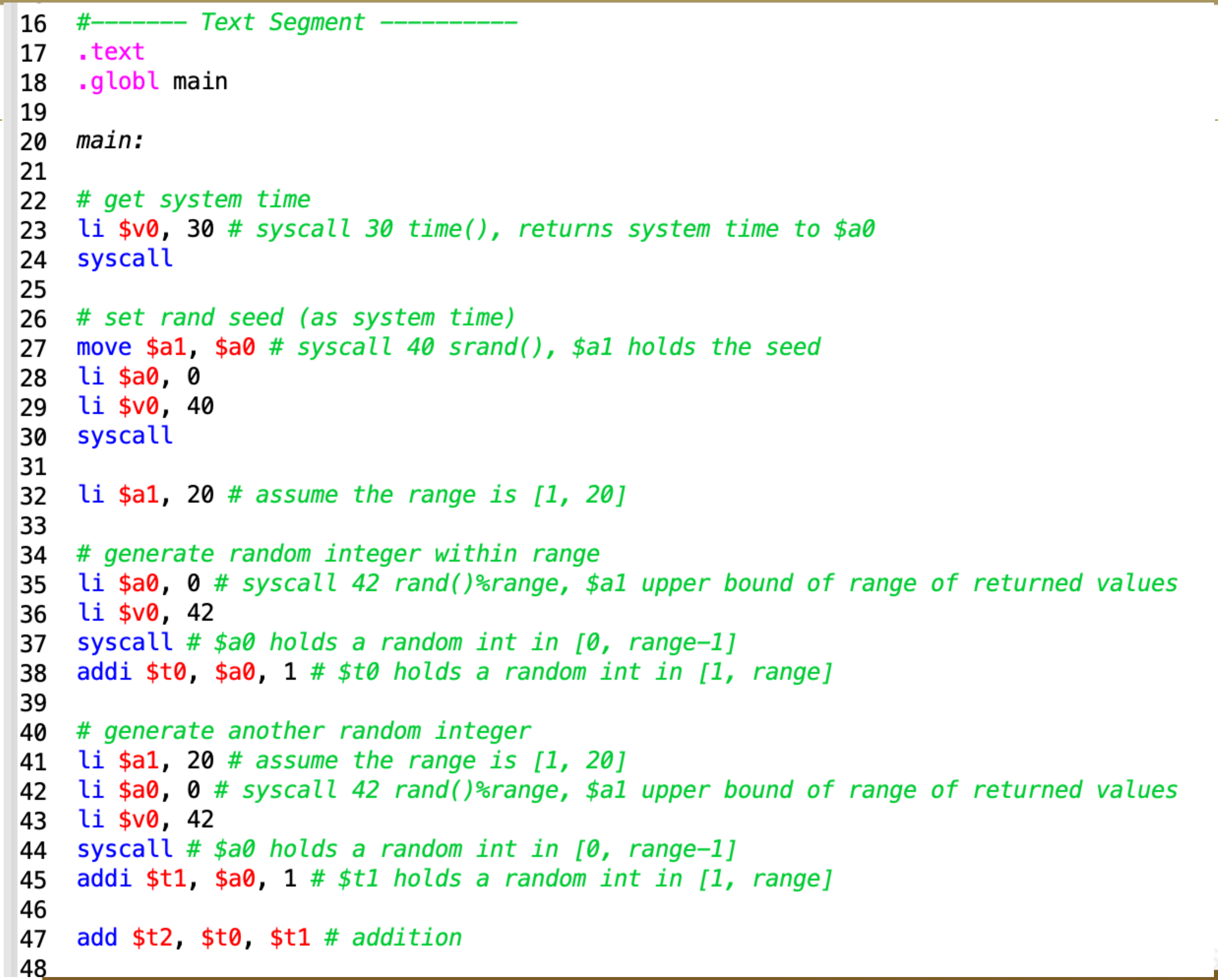


```
# Add two numbers
# $t0: 1st number
# $t1: 2nd number
# $v0: syscall id and return value
# $a0: syscall argument

.text
.globl main

main:

# read the 1st nunber from keyboard
li $v0, 5 # read int
syscall # the int is in $v0
move $t0, $v0 # pseudo instruction

# do similiar for the 2nd nunber
li $v0, 5 # read int
syscall # the int is in $v0
move $t1, $v0 # pseudo instruction

add $t2, $t0, $t1 # addition

# print int
move $a0, $t2
li $v0, 1 # value in $a0 is printed
syscall

# exit
li $v0, 10
syscall

```

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

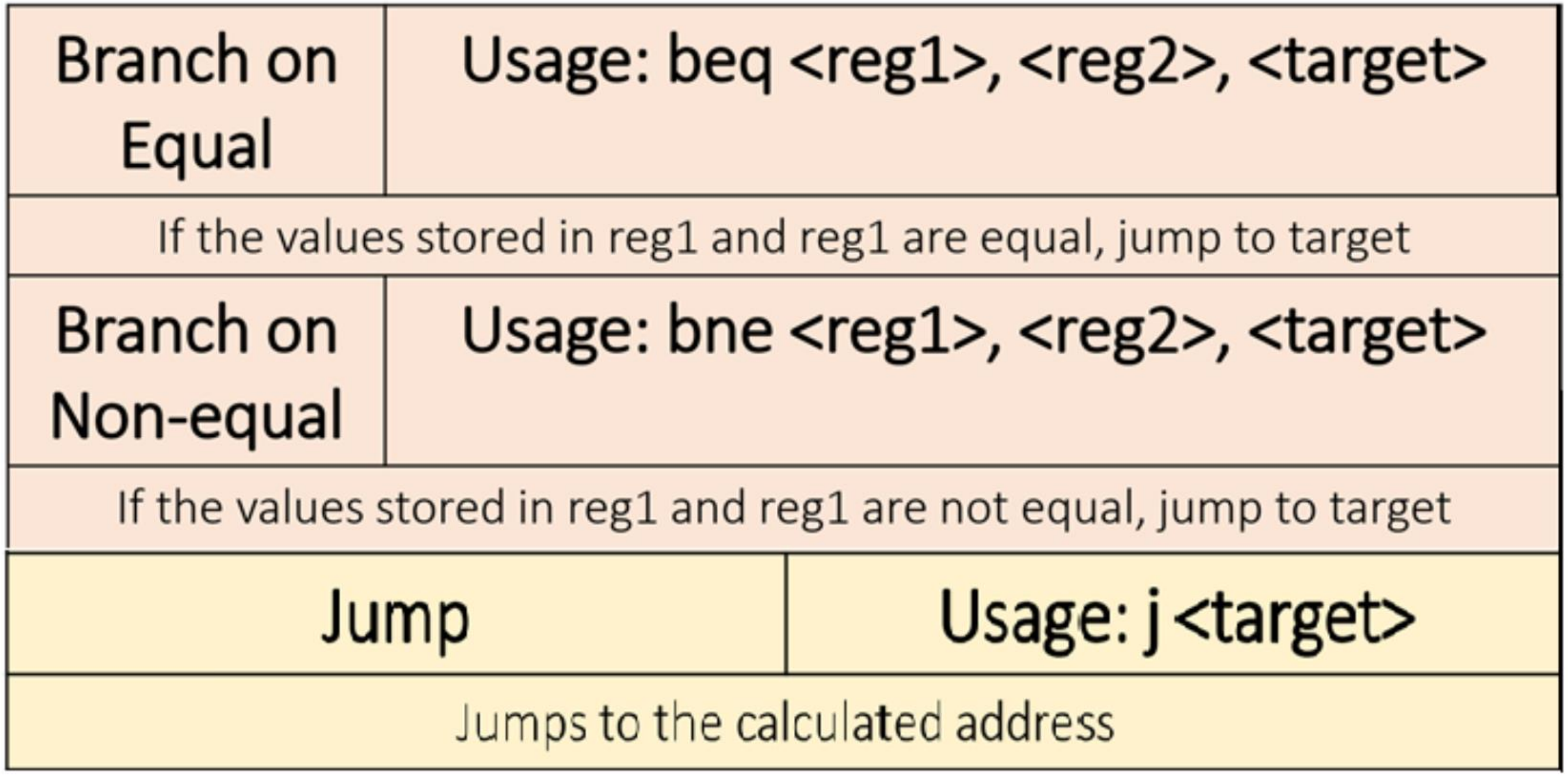
Example: Sum of Two Random Generated Integers

- **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**



```
#------- Text Segment ----------
.text
.globl main

main:

# get system time
li $v0, 30 # syscall 30 time(), returns system time to $a0
syscall

# set rand seed (as system time)
move $a1, $a0 # syscall 40 srand(), $a1 holds the seed
li $a0, 0
li $v0, 40
syscall

li $a1, 20 # assume the range is [1, 20]

# generate random integer within range
li $a0, 0 # syscall 42 rand()%range, $a1 upper bound of range of returned values
li $v0, 42
syscall # $a0 holds a random int in [0, range-1]
addi $t0, $a0, 1 # $t0 holds a random int in [1, range]

# generate another random integer
li $a1, 20 # assume the range is [1, 20]
li $a0, 0 # syscall 42 rand()%range, $a1 upper bound of range of returned values
li $v0, 42
syscall # $a0 holds a random int in [0, range-1]
addi $t1, $a0, 1 # $t1 holds a random int in [1, range]

add $t2, $t0, $t1 # addition

```

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

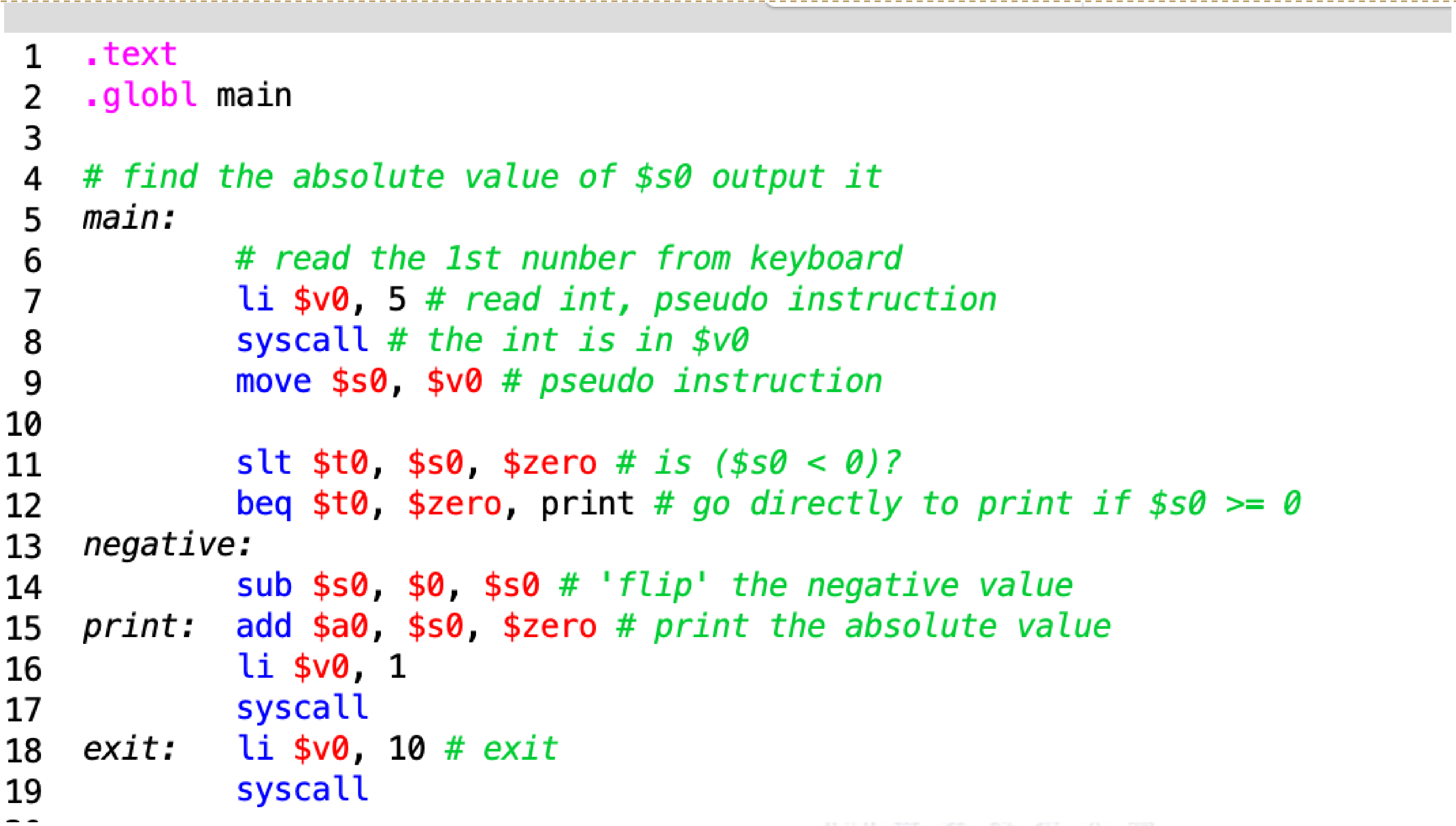
Set on Less Than	Usage: slt <reg 1>, <reg 2>, <reg 3>
Register reg1 is set to 1 if the value in reg2 is less than the value in reg3; otherwise, register reg1 is set to 0	
Set on Less Than Immediate	Usage: slti <reg 1>, <reg 2>, <immediate>
Register reg1 is set to 1 if the value in reg2 is less than the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

Example: Absolute Value

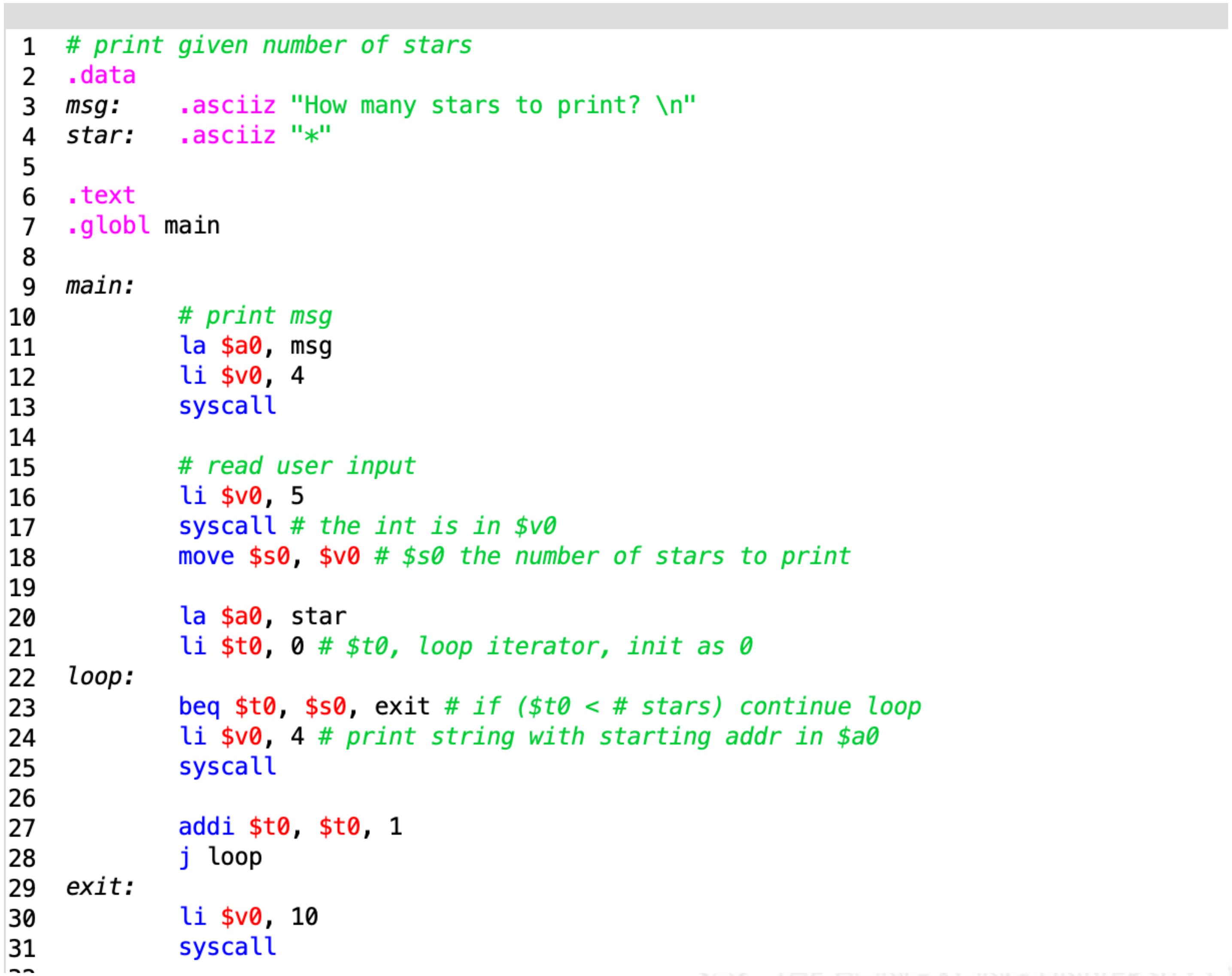


```
.text
.globl main

# find the absolute value of $s0 output it
main:
        # read the 1st nunber from keyboard
        li $v0, 5 # read int, pseudo instruction
        syscall # the int is in $v0
        move $s0, $v0 # pseudo instruction

        slt $t0, $s0, $zero # is ($s0 < 0)?
        beq $t0, $zero, print # go directly to print if $s0 >= 0
negative:
        sub $s0, $0, $s0 # 'flip' the negative value
print:  add $a0, $s0, $zero # print the absolute value
        li $v0, 1
        syscall
exit:   li $v0, 10 # exit
        syscall
```

Example: Add Even Integers

```
.text
.globl main

main:
        li $s0, 100 # add up all even numbers in range [1, $s0], li pseudo instruction
        li $t0, 1 # $t0, loop iterator
        li $t1, 0 # $t1, sum

loop:
        slt $t2, $s0, $t0 # if (i <= 100), which is equivalent if !(100 < i)
        bne $t2, $zero, exit

        andi $t3, $t0, 1 # $t3 either 0000...000 (when $t0 is even) or 0000...001 (when $t0 is odd)

        bne $t3, $0, next_iteration # if i is odd, skip the addition
        add $t1, $t1, $t0 # sum += i if i is even

next_iteration:
        addi $t0, $t0, 1 # i++
        j loop

exit:
        li $v0, 10
        syscall

```

Example: Simple for-Loop



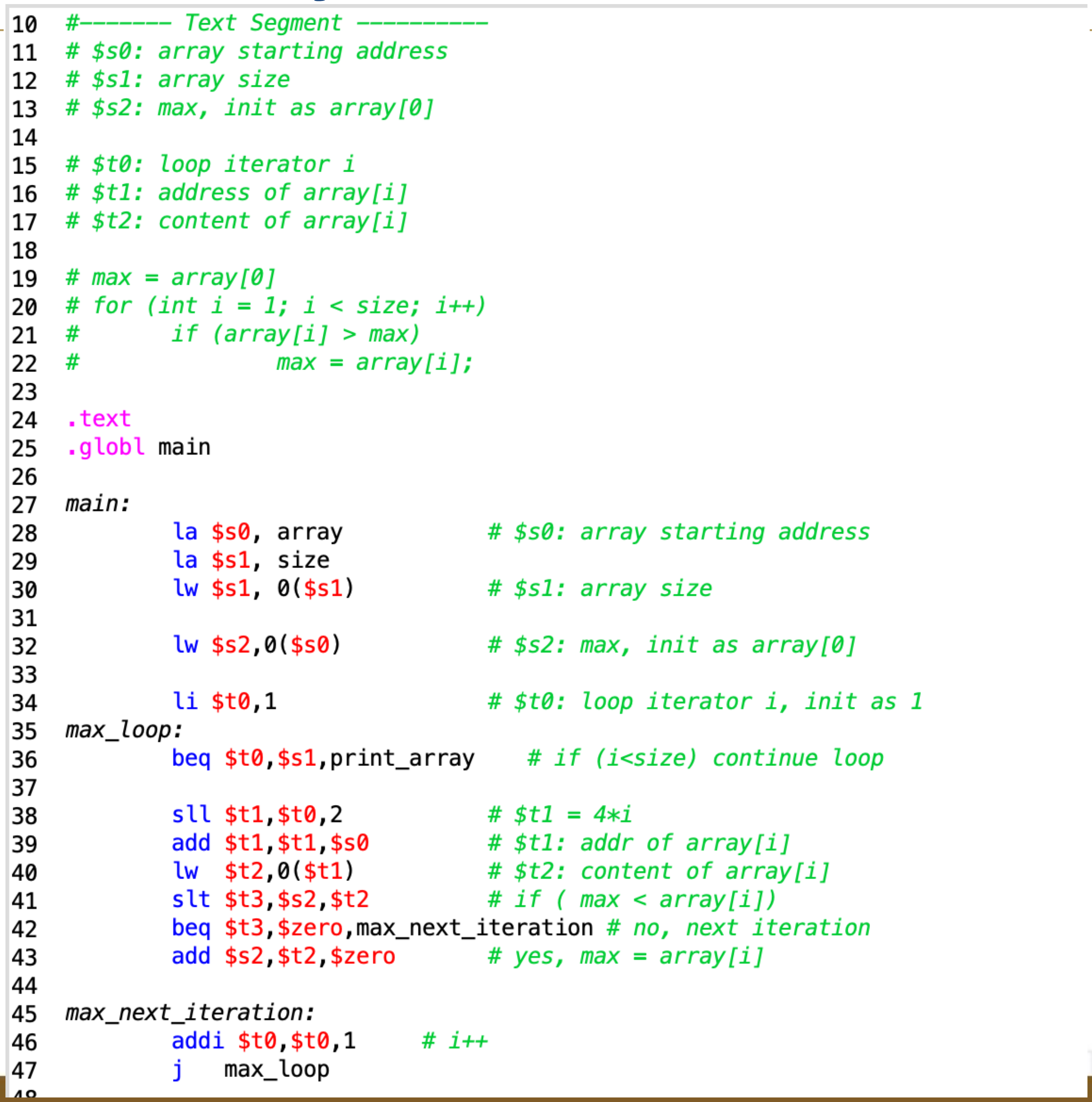
```
# print given number of stars
.data
msg:    .asciiz "How many stars to print? \n"
star:   .asciiz "*"

.text
.globl main

main:
        # print msg
        la $a0, msg
        li $v0, 4
        syscall

        # read user input
        li $v0, 5
        syscall # the int is in $v0
        move $s0, $v0 # $s0 the number of stars to print

        la $a0, star
        li $t0, 0 # $t0, loop iterator, init as 0
loop:
        beq $t0, $s0, exit # if ($t0 < # stars) continue loop
        li $v0, 4 # print string with starting addr in $a0
        syscall

        addi $t0, $t0, 1
        j loop
exit:
        li $v0, 10
        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);
```

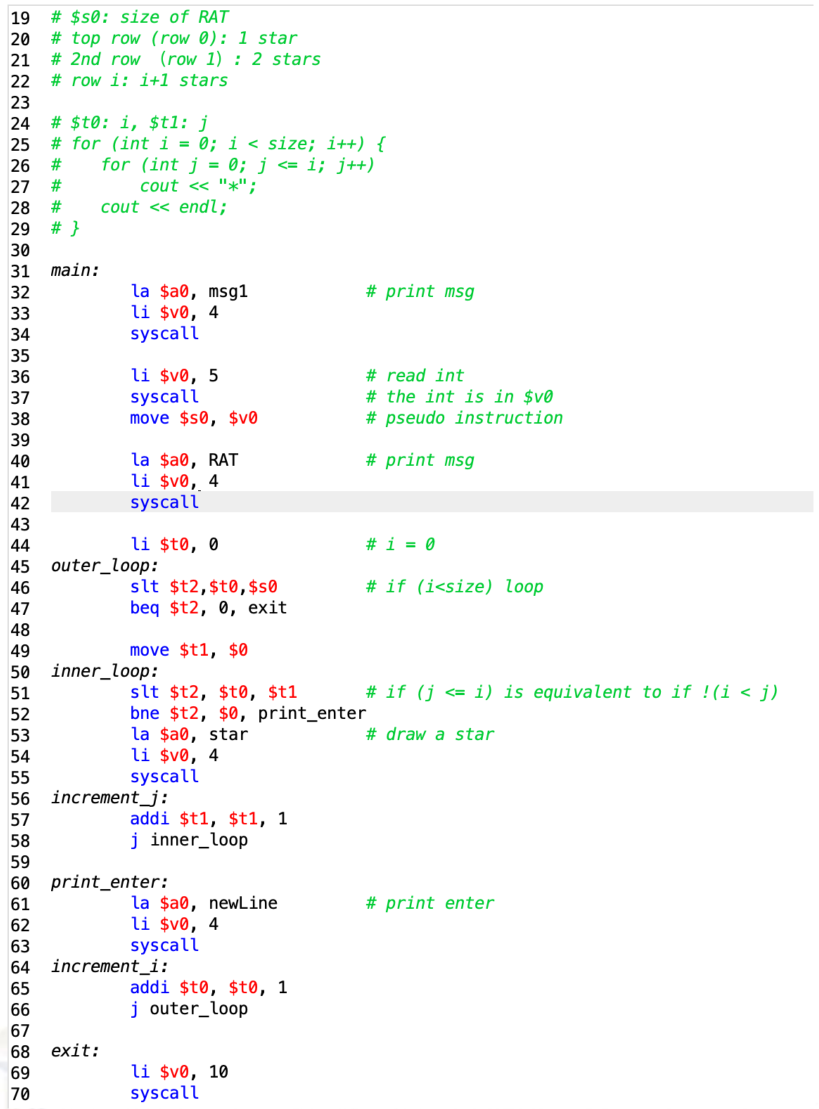
Example: Max in an Array



```
#------- Text Segment ----------
# $s0: array starting address
# $s1: array size
# $s2: max, init as array[0]

# $t0: loop iterator i
# $t1: address of array[i]
# $t2: content of array[i]

# max = array[0]
# for (int i = 1; i < size; i++)
#       if (array[i] > max)
#               max = array[i];

.text
.globl main

main:
        la $s0, array           # $s0: array starting address
        la $s1, size
        lw $s1, 0($s1)          # $s1: array size

        lw $s2,0($s0)           # $s2: max, init as array[0]

        li $t0,1                # $t0: loop iterator i, init as 1
max_loop:
        beq $t0,$s1,print_array    # if (i<size) continue loop

        sll $t1,$t0,2           # $t1 = 4*i
        add $t1,$t1,$s0         # $t1: addr of array[i]
        lw  $t2,0($t1)          # $t2: content of array[i]
        slt $t3,$s2,$t2         # if ( max < array[i])
        beq $t3,$zero,max_next_iteration # no, next iteration
        add $s2,$t2,$zero       # yes, max = array[i]

max_next_iteration:
        addi $t0,$t0,1     # i++
        j   max_loop
```

Example: Nested for-Loop

- Use proper labels
- Maintain a good 'variable-register' mapping table



```
# $s0: size of RAT
# top row (row 0): 1 star
# 2nd row  (row 1) : 2 stars
# row i: i+1 stars

# $t0: i, $t1: j
# for (int i = 0; i < size; i++) {
#    for (int j = 0; j <= i; j++)
#        cout << "*";
#    cout << endl;
# }

main:
        la $a0, msg1            # print msg
        li $v0, 4
        syscall

        li $v0, 5               # read int
        syscall                 # the int is in $v0
        move $s0, $v0           # pseudo instruction

        la $a0, RAT             # print msg
        li $v0, 4
        syscall

        li $t0, 0               # i = 0
outer_loop:
        slt $t2,$t0,$s0         # if (i<size) loop
        beq $t2, 0, exit

        move $t1, $0
inner_loop:
        slt $t2, $t0, $t1       # if (j <= i) is equivalent to if !(i < j)
        bne $t2, $0, print_enter
        la $a0, star            # draw a star
        li $v0, 4
        syscall
increment_j:
        addi $t1, $t1, 1
        j inner_loop

print_enter:
        la $a0, newLine         # print enter
        li $v0, 4
        syscall
increment_i:
        addi $t0, $t0, 1
        j outer_loop

exit:
        li $v0, 10
        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;
}
```

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--;
}
```

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];
}
```