# CS224

Section 01

Fall 2019

Lab No. 1

Oğuz Kaan İmamoğlu / 21702233

**1.**

```
# read n from console
la $a0,prompt1 # print string before result
li $v0,4
syscall

li $v0 5
syscall
move $t0 $v0

# allocate dynamic memory
sll $a0 $v0 2    # sll performs $a0 = $v0 x 2^2
li $v0 9    #9 is the system code for service(sbrk) whoes work is
syscall     #to allocate dynamic memory

move $t1 $zero
move $t2 $v0
li $s1, 0
li $s2, 0
move $t4, $t2

loop:
  bge $t1 $t0 end

  la $a0,prompt2          # print string before result
  li $v0,4
  syscall

  # read in and store int
  li $v0 5
  syscall
  sw $v0 0($t2)

  addi $t1 $t1 1
  addi $t2 $t2 4
  j loop

end:

print:
        slt  $t6, $s2, $t0 # counter less than result
        beq  $t6, $0, printend

        lw $t3 0($t4)
        move $a0,$t3
        li $v0,1
        syscall

        addi $t4, $t4, 4

        addi $s2, $s2, 1
        j print

printend:

printreverse:

        slt  $t6, $s1, $t0 # counter less than result
        beq  $t6, $0, finish

        subi $t2 $t2 4
        lw $t3 0($t2)
        move $a0,$t3
        li $v0,1
        syscall


        addi $s1, $s1, 1
        j printreverse

finish:

        .data
prompt1:.asciiz "Enter  array size: "
prompt2:.asciiz "Enter array element: "
```

## 2.

```
# read n from console
la $a0,prompt1 # print string before
li $v0,4
syscall

li $v0 5
syscall
move $t0 $v0

# allocate dynamic memory
```

```
sll $a0 $v0 2    # sll performs $a0 = $v0 x 2^2
li $v0 9    #9 is the system code for service(sbrk) whoes work is
syscall     #to allocate dynamic memory

move $t1 $zero
move $t2 $v0

li $t3 0 #counter

li $s5 2 #this holds 2

div $s0, $t0, $s5 # s0 is limit


move $t6 $t2 # t6 holds first value

loop:
   bge $t1 $t0 end

   la $a0,prompt2 # print string before
   li $v0,4
   syscall

   # read in and store int
   li $v0 5
   syscall
   sw $v0 0($t2)

   addi $t1 $t1 1
   addi $t2 $t2 4
   j loop
```

```
end:

while:
        bge $t3 $s0 end2

        subi $t2, $t2,4

        lw $t4  0($t6)
        lw $t5  0($t2)

        bne $t5 $t4 printno

        addi $t6, $t6, 4


        addi $t3, $t3, 1


        j while
end2:

        la $a0,yes          # print string before result
        li $v0,4
        syscall

        j theend


printno:
        la $a0,no       # print string before result
        li $v0,4
        syscall

theend:


        .data
result: .space 12
yes:    .asciiz "Yes"
no:     .asciiz "No"
prompt1:.asciiz "Enter  array size: "
prompt2:.asciiz "Enter array element: "
```

**3.**

```
        .text
        .globl __start

__start:

        la $a0,prompt   # print prompt on terminal
        li $v0,4
        syscall

        li $v0,5 # syscall 5 reads an integer
        syscall
        move $s0,$v0    # $s0 holds number

        la $a0,prompt   # print prompt on terminal
        li $v0,4
        syscall

        li $v0,5  # syscall 5 reads an integer
        syscall
        move $s1,$v0    # $s1 holds number

        li $s2,0  # $s2 will be the division
        li $t3,0  # $s3 will be the remainder
        add $s4,$0,$s0 # $s4 is to remember initial value


loop:

        #slt  $t5, $s1, $s0 # counter less than result
        #beq  $t5, $0, done

        sub $s0,$s0,$s1
        addi $s2,$s2,1
        blt $s0,$s1 done


        j loop

done:
    la $a0,ans1# print string before result
        li $v0,4
        syscall

        move $a0,$s2
        li $v0,1
        syscall
```

```
          la $a0,ans2        # print string before result
          li $v0,4
          syscall

          move $a0,$s0
          li $v0,1
          syscall


                              .data
prompt:             .asciiz "Enter  number: "
ans1:     .asciiz "Division is "
ans2:     .asciiz "\nRemainder is "
result:   .space 12
```

## 4.

```
add       $t0, $t1, $t2      00000001001010100100000000100000
0x012a4020


addi      $s0, $s3, 15       00100010011100000000000000001111
```

0x2270000f

mult $a0, $a1 00000000100001010000000000011000

0x00850018

sw $t1, 8($t2) 10101101010010010000000000001000

0xad490008

lw $t2, 8($t1) 10001101001010100000000000001000

0x8d2a0008

## 5.

a. Symbolic machine instruction: sub $s0,$s0,$s1
addi $s2,$s2,1

b. Machine instruction: mult $a0, $a1 00000000100001010000000000011000
sw $t1, 8($t2) 10101101010010010000000000001000

c. Assembler directive: .globl __start
.text

d. Pseudo instruction:

| **Name** | **Assembly syntax** | **Expansion** |
|---|---|---|
| move | move $t, $s | or $t, $s, $zero |
| clear | clear $t | or $t, $zero, $zero |