CS224

Section No: 1

Fall 2019

Lab No 1

Fatih Sevban Uyanik / 21602486

Q1)

.data

array: .space 80

reversed\_array: .space 80

array\_size: .word 20

array\_length: .word 0

request\_aray\_length: .asciiz "Please enter the length of the array: "

request\_aray\_item\_1: .asciiz "Please enter the "

request\_aray\_item\_2: .asciiz " item: "

content\_of\_array: .asciiz "Array Content: "

add\_space: " "

new\_line: "\n"

.text

addi $t0, $zero, 0

addi $t1, $zero, 0

lw $t2, array\_size

# requesting length

addi $v0, $zero, 4

la $a0, request\_aray\_length

syscall

# getting user input

addi $v0, $zero, 5

syscall

# storing length

add $t3, $zero, $v0

sw $t3, array\_length

j check\_1

# requesting items

for\_1: addi $t0, $t0, 1

addi $v0, $zero, 4

la $a0, request\_aray\_item\_1

syscall

# printing an integer

addi $v0, $zero, 1

addi $a0, $t0, 0

syscall

# requesting item

addi $v0, $zero, 4

la $a0, request\_aray\_item\_2

syscall

# getting user input

addi $v0, $zero, 5

syscall

sw $v0, array($t1)

addi $t1, $t1, 4

check\_1: bne $t0, $t3, for\_1

addi $v0, $zero, 4

la $a0, content\_of\_array

syscall

addi $t0, $zero, 0

addi $t1, $zero, 0

j check\_2

# Displaying array

for\_2: addi $t0, $t0, 1

lw $t5, array($t1)

addi $t1, $t1, 4

# printing an integer

addi $v0, $zero, 1

addi $a0, $t5, 0

syscall

addi $v0, $zero, 4

la $a0, add\_space

syscall

check\_2: bne $t0, $t3, for\_2

# Reversing array

addi $v0, $zero, 4

la $a0, new\_line

syscall

addi $t0, $zero, 0

addi $t1, $zero, 0

lw $t4, array\_length

addi $t4, $t4, -1

mul $t4, $t4, 4

j check\_3

for\_3: addi $t0, $t0, 1

lw $t5, array($t1)

addi $t1, $t1, 4

sw $t5, reversed\_array($t4)

addi $t4, $t4, -4

check\_3: bne $t0, $t3, for\_3

addi $t0, $zero, 0

addi $t1, $zero, 0

j check\_4

# copying reversed array

# contents to actual array

for\_4: addi $t0, $t0, 1

lw $t5, reversed\_array($t1)

sw $t5, array($t1)

addi $t1, $t1, 4

check\_4: bne $t0, $t3, for\_4

# Displaying array

addi $v0, $zero, 4

la $a0, content\_of\_array

syscall

addi $t0, $zero, 0

addi $t1, $zero, 0

j check\_5

for\_5: addi $t0, $t0, 1

lw $t5, reversed\_array($t1)

addi $t1, $t1, 4

# printing an integer

addi $v0, $zero, 1

addi $a0, $t5, 0

syscall

addi $v0, $zero, 4

la $a0, add\_space

syscall

check\_5: bne $t0, $t3, for\_5

addi $v0, $zero, 4

la $a0, new\_line

syscall

Q2)

.data

array: .space 80

array\_size: .word 20

array\_length: .word 0

request\_aray\_length: .asciiz "Please enter the length of the array: "

request\_aray\_item\_1: .asciiz "Please enter the "

request\_aray\_item\_2: .asciiz " item: "

content\_of\_array: .asciiz "Array Content: "

not\_palindrome: .asciiz "The array is not a palindrome."

is\_palindrome: .asciiz "The array is a palindrome."

add\_space: " "

new\_line: "\n"

.text

addi $t0, $zero, 0

addi $t1, $zero, 0

lw $t2, array\_size

# requesting length

addi $v0, $zero, 4

la $a0, request\_aray\_length

syscall

# getting user input

addi $v0, $zero, 5

syscall

# storing length

add $t3, $zero, $v0

sw $t3, array\_length

j check\_1

# requesting items

for\_1: addi $t0, $t0, 1

addi $v0, $zero, 4

la $a0, request\_aray\_item\_1

syscall

# printing an integer

addi $v0, $zero, 1

addi $a0, $t0, 0

syscall

# requesting item

addi $v0, $zero, 4

la $a0, request\_aray\_item\_2

syscall

# getting user input

addi $v0, $zero, 5

syscall

sw $v0, array($t1)

addi $t1, $t1, 4

check\_1: bne $t0, $t3, for\_1

addi $v0, $zero, 4

la $a0, content\_of\_array

syscall

addi $t0, $zero, 0

addi $t1, $zero, 0

j check\_2

# Displaying array

for\_2: addi $t0, $t0, 1

lw $t5, array($t1)

addi $t1, $t1, 4

# printing an integer

addi $v0, $zero, 1

addi $a0, $t5, 0

syscall

addi $v0, $zero, 4

la $a0, add\_space

syscall

check\_2: bne $t0, $t3, for\_2

addi $v0, $zero, 4

la $a0, new\_line

syscall

addi $s0, $zero, 0

lw $s1, array\_length

addi $s1, $s1, -1

la $t0, array

mul $t1, $s1, 4

add $t1, $t1, $t0

j check\_3

# Displaying array

for\_3: addi $s0, $s0, 1

addi $s1, $s1, -1

lw $t4, 0($t0)

lw $t5, 0($t1)

addi $t0, $t0, 4

addi $t1, $t1, -4

bne $t4, $t5, non\_palindrome

check\_3: slt $t3, $s0, $s1

bne $t3, $zero, for\_3

addi $v0, $zero, 4

la $a0, is\_palindrome

syscall

j exit\_1

non\_palindrome: addi $v0, $zero, 4

la $a0, not\_palindrome

syscall

exit\_1:

Q3)

.data

dividend: .word 0

divisor: .word 0

request\_dividend: .asciiz "Please enter the dividend number : "

request\_divisor: .asciiz "Please enter the divisor number : "

represent\_quotient: .asciiz "Quotient : "

represent\_remainder: .asciiz "Remainder : "

new\_line: "\n"

.text

# requesting dividend

addi $v0, $zero, 4

la $a0, request\_dividend

syscall

# getting dividend

addi $v0, $zero, 5

syscall

# storing dividend

add $t0, $zero, $v0

sw $t0, dividend

# requesting divisor

addi $v0, $zero, 4

la $a0, request\_divisor

syscall

# getting divisor

addi $v0, $zero, 5

syscall

# storing divisor

add $t1, $zero, $v0

sw $t1, divisor

addi $t2, $zero, 0

while\_1: slt $t3, $t0, $t1

bne $t3, $zero, show\_result

sub $t0, $t0, $t1

addi $t2, $t2, 1

j while\_1

show\_result:

addi $v0, $zero, 4

la $a0, represent\_quotient

syscall

addi $v0, $zero, 1

addi $a0, $t2, 0

syscall

addi $v0, $zero, 4

la $a0, new\_line

syscall

addi $v0, $zero, 4

la $a0, represent\_remainder

syscall

addi $v0, $zero, 1

addi $a0, $t0, 0

syscall

Q4)

**add $t0, $t1, $t2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 9 | 10 | 8 | 0 | 32 |

Binary: 000000 01001 01010 01000 00000 100000

Hex: 0x01294020

**addi $s0, $s3, 15**

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | 19 | 16 | 15 |

Binary: 001000 10011 10000 0000000000001111

Hex: 0x2270000F

**mult $a0, $a1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 24 | 4 | 5 | 0 | 0 | 0 |

Binary: 011000 00100 00101 00000 00000 000000

Hex: 0x60850000

**sw $t1, 8($t2)**

|  |  |  |  |
| --- | --- | --- | --- |
| 43 | 10 | 9 | 8 |

Binary: 101011 01010 01001 0000000000001000

Hex: 0xAD490008

**lw $t2, 8($t1)**

|  |  |  |  |
| --- | --- | --- | --- |
| 35 | 9 | 10 | 8 |

Binary: 100011 01001 01010 0000000000001000

Hex: 0x8D2A0008

Q5)

**Symbolic machine instruction:**

The instructions of assembly languages are called Symbolic machine instructions. It makes the machine code more human readable and programmable. The assembler converts these symbolic machine instructions into machine instructions.

**- sw $s1, 0($s2)**

stores the value in $s1 into the memory adressed stored in $s2. In this case, sw and $s1 are examples of symbolic representations.

**- sub $a0, $a1, $a2**

subtractes the register value in $a1 from $a2 and assigns it to register to $a0. In this case, sub and $a1 are examples of symbolic representations.

**Machine instruction:**

Machine instruction is the instruction set that the machine can understand.Every machine instruction starts with an op code that reflects which operation should be done and the remainings instructions are the needed data to perform this operation.

**- op code 00000 and func 100000**

indicates that the machine instruction preforms addition.

**- op code 101011**

indicates that the machine instruction preforms storing data to memory.

**Assembler directive:**

Assembler directives are set of Instructions that directs the assembler to do certain tasks.

**- ELSE**

Executes the program if previous if statement is wrong.

**- EQU**  
Assigns a value from one variable to another variable.

**Pseudo Instructions:**

These instructions are a shortcut of Symbolic machine instructions. These instructions enables to code complex tasks in a neater and organised way.

**- move $a0, $t2**

This assigns the value of $t2 to $a0 and is a shortcut of “addi $a0, $t2, 0”

**- li $t4, 10**

This assigns the value of 10 to $t0 immediately and is a shortcut of “addi $t4, $0, 10”