Department of Computer Science & Engineering, University of Dhaka

CSE-3113
Microprocessor and Assembly Language Lab

Assignment on Loop and Array

Submitted By:
Mirajul Mohin, Roll: FH-28

Date of Submission: 20 March 2018

Submitted to:

Mr. Tamal Adhikary, Lecturer, Dept. of CSE, DU
Dr. Md. Mustafizur Rahman, Professor, Dept. of CSE, DU

ARRAY:

Task-1:

```
segment .data
        fmt: db "%d",0
        fmt2: db "%d ",10,0
        fmt3: db "%d ",0

segment .bss
        a: resq 255
        b: resq 30
        cnt: resq 1
        d: resq 30

segment .text
global main
extern printf
extern scanf

main:
        push RBP
        xor RCX,RCX
        mov RCX,20
        mov RBX,0

for:
        mov RAX,0
        mov RDI,fmt
        mov RSI,d

        push RBX
        push RCX
        call scanf
        mov RAX,[d]
        pop RCX
        pop RBX

        mov [a+RBX*8],RAX
        inc RBX
        add RAX,[cnt]
        mov [cnt],RAX
        loop for

        xor RAX,RAX
        mov RDI,fmt2
        mov RSI,[cnt]
        call printf
        xor RCX,RCX

print:
```

```
        mov RBX,[a+8*RCX]
        mov RDI,fmt3
        mov RSI,RBX
        mov RAX,0
        push RCX
        call printf
        pop RCX
        inc RCX
        cmp RCX,20
        jne print

        xor RAX,RAX
        pop RBP
        ret
```

Task-2:

```
segment .data
        fmt: db "%d",0
        fmt2: db "%d ",10,0
        fmt3: db "%d ",0

segment .bss
        a: resq 255
        b: resq 30
        cnt: resq 1
        d: resq 30

segment .text
global main
extern printf
extern scanf

main:
        push RBP
        xor RCX,RCX
        mov RCX,10
        mov RBX,0

for:
        mov RAX,0
        mov RDI,fmt
        mov RSI,d

        push RBX
        push RCX
        call scanf
        mov RAX,[d]
        pop RCX
        pop RBX
```

```
        mov [a+RBX*8],RAX
        inc RBX
        loop for

        mov RAX,0
        mov RDI,fmt
        mov RSI,d
        call scanf
        mov RAX,[d]

        mov RCX,0

print:
        mov R8,[a+8*RAX]
        mov RDI,fmt2
        mov RSI,R8
        mov RAX,0
        call printf

        xor RAX,RAX
        pop RBP
        ret
```

Task-3:

```
segment .data
fmt: db "%d",0
fmt2: db "%d ",10,0
fmt3: db "%d ",0
fmt5: db "No first and last odd number",10,0
fmt6: db "First odd element %d and there is no last element",10,0

segment .bss
a: resq 255
cnt: resq 255
b: resq 1
c: resq 1
d: resq 1

segment .text
global main
extern printf
extern scanf

main:
push RBP

xor RCX,RCX
mov R9,0
```

```
for:
        push RCX
        xor RAX,RAX
        mov RDI,fmt
        mov RSI,d
        call scanf
        pop RCX
        mov RAX,[d]
        mov [a+RCX*8],RAX
        inc RCX
        cmp RCX,10
        jne for



        mov RAX,0
        mov RCX,0
        mov R8,0

print:
        mov RBX,[a+8*RCX]
        and RBX,1
        cmp RBX,0
        je even
        jmp odd


odd:

        mov RBX,[a+8*RCX]
        mov [cnt+8*R8],RBX
        INC R8
        INC RCX
        cmp RCX,10
        je _p
        jmp print


even:
        inc RCX
        cmp RCX,10
        jne print
        jmp _p

_p:
        mov RAX,0
        mov RCX,0

        cmp R8,0
```

```
        je no
        cmp R8,1
        je one

        mov RDI,fmt2
        mov RSI,[cnt+RCX*8]
        push R8
        call printf

        pop R8
        DEC R8
        mov RCX,R8
        mov RAX,0
        mov RDI,fmt2
        mov RSI,[cnt+8*RCX]
        call printf
        jmp exit

no:
        mov RDI,fmt5
        call printf
        jmp exit

one:
        mov RCX,0
        mov RDI,fmt6
        mov RSI,[cnt+8*RCX]
        call printf
        jmp exit

exit:
xor RAX,RAX
pop RBP
ret
```

Task-4

```
segment .data

a: dq 0
i: dq 0
tmp: dq 0


cnt: dq 0

fmt: dq "%lld ",10,0
fmt_put: dq "%lld",0

segment .bss
```

```
ara resq 21
array resq 21

segment .text
global main
extern printf
extern scanf

main:
push RBP

mov RAX,0
mov RBX,0
mov RCX,0

input:

        cmp RCX,5
        je reset
        push RCX
        mov RAX,0
    mov RDI,fmt_put
        mov RSI,a
        call scanf
    mov RAX,[a]
        pop RCX
        mov [ara+RCX*8],RAX
        inc RCX
    jmp input


reset:
    mov RAX,0
    mov RCX,0
    mov R10,0

outer:
    cmp RCX,5
    jge zero
    mov [cnt],RCX
    mov R10,[ara+RCX*8]

inner:
    inc RCX
    mov [tmp],RCX
    cmp RCX,5
    jge checker
    mov RCX,[tmp]
    cmp R10,[ara+RCX*8]
```

```
        jle inner
        mov RCX,[tmp]
        xchg R10,[ara+RCX*8]
        jmp inner

checker:
        mov RCX,[cnt]
        mov [ara+RCX*8],R10
        inc RCX
        jmp outer

zero:
        mov RAX,0
        mov RCX,0
           mov R8,0


print:
        mov RAX,[ara+2*8]
        mov RDI,fmt
        mov RSI,RAX
        call printf



Finish:
        mov RAX,0
        pop RBP
        ret
```

Task-6

```
segment .data

a: dq 0
i: dq 0
tmp: dq 0


cnt: dq 0

fmt: dq "%lld ",10,0
fmt_put: dq "%lld",0

segment .bss
ara resq 21

segment .text
global main
extern printf
```

```
extern scanf

main:
push RBP

mov RAX,0
mov RBX,0
mov RCX,0

input:

        cmp RCX,5
        je reset
        push RCX
        mov RAX,0
    mov RDI,fmt_put
        mov RSI,a
        call scanf
    mov RAX,[a]
        pop RCX
        mov [ara+RCX*8],RAX
        inc RCX
    jmp input


reset:
    mov RAX,0
    mov RCX,0
    mov R10,0

outer:
    cmp RCX,5
    jge zero
    mov [cnt],RCX
    mov R10,[ara+RCX*8]

inner:
    inc RCX
    mov [tmp],RCX
    cmp RCX,5
    jge checker
    mov RCX,[tmp]
    cmp R10,[ara+RCX*8]
    jle inner
    mov RCX,[tmp]
    xchg R10,[ara+RCX*8]
    jmp inner

checker:
```

```
        mov RCX,[cnt]
        mov [ara+RCX*8],R10
        inc RCX
        jmp outer

zero:
        mov RAX,0
        mov RCX,0

print:
        cmp RCX,5
        je Finish
        mov RAX,[ara+RCX*8]
        inc RCX
        mov [cnt],RCX
        mov RDI,fmt
        mov RSI,RAX
        call printf
        mov RCX,[cnt]
        jmp print


Finish:
        mov RAX,0
        pop RBP
        ret
```

Task-7

```
segment .data

no_of_vowels:           dq      0
no_of_conso:        dq      0
d:          dq      128

cnt:        dq      0
fmt_in:     db      "%s",0
fmt         db      "%d",10,0
fmt_out1:   db      "Given input: %s",10,0
fmt_out4:   db      "Vowels: %lld ",0
fmt_out3:   db      "Consonants: %lld ",10,0

segment .bss
array resq 21

segment .text
global main
extern printf
extern scanf
extern gets
```

```
main:
        push RBP

        mov RDI,array
        call gets

        mov RCX,0
        mov RBX,0

CNT:
        cmp [array+RCX],RBX
        jz Print

        mov AL,[array+RCX]

        cmp AL,'A'
        jz vowels

        cmp AL,'E'
        jz vowels

        cmp AL,'I'
        jz vowels

        cmp AL,'O'
        jz vowels

        cmp AL,'U'
        jz vowels

        cmp AL,' '
        jnz consonant

        jmp increment

vowels:
        inc qword[no_of_vowels]
        jmp increment

consonant:
        inc qword[no_of_conso]

increment:
        inc RCX
        jmp CNT

Print:
```

```
        mov RDI,fmt_out1
        mov RSI,array
        call printf

        mov RBX,[no_of_vowels]
        add RBX,4
        mov RDI,fmt_out4
        mov RSI,RBX
        call printf

        xor RBX,RBX

        mov RBX,[no_of_conso]
        add RBX,6
        mov RDI,fmt_out3
        mov RSI,RBX
        call printf

        pop RBP
        ret
```

LOOP:

Task-1

```
segment .data
        prime: db "prime" ,10 , 0
        non_prime: db "non prime",10,0
        fmt: dq "%s", 10, 0
        fmt_scan: dq "%d %d",10,0
        fmt_value: dq "%d ",10,0
segment .bss
        a: resq 1
        b: resq 1
        array: resq 255

segment .text
        global main
        extern printf
        extern scanf

main:
        push RBP

        mov RDI,fmt_scan
        mov RSI,a
        mov RDX,b
        mov RAX,0
        call scanf

        mov RAX,[a]
```

```
        mov RBX,[b]

        mov RCX,RBX
        sub RCX,RAX
        inc RCX

        xor RDX,RDX

        xor R9,R9
        mov R9,RAX
        add R9,-2

        mov R10,0
        mov R8,2
        mov R11,0


mPrint:
        push RCX

        push RAX

        xor RDX,RDX
        div R8
        cmp RDX,0

        je nonPrime

        pop RAX
        pop RCX
        inc R8
        inc r10

        cmp R10,R9
        jne mPrint



y_prime:
        mov [array+8*R11],RAX
        inc R11
        inc RAX
        mov R8,2
        mov R9,RAX
        add R9,-2
        mov R10,0

        dec RCX
        cmp RCX,0
```

```
        jne mPrint
        jmp finish


nonPrime:
        pop RAX
        pop RCX

        inc RAX
        mov R8,2
        mov R9,RAX
        add R9,-2
        mov R10,0

        loop mPrint


finish:
        mov RCX,0

exit:
        mov RDI,fmt_value
        mov RSI,[array+8*RCX]
        inc RCX
        push RCX
        push R11
        mov RAX,0
        call printf
        pop R11
        pop RCX
        cmp R11, RCX
        jne exit
        pop RBP
        ret
```

Task-2

```
segment .data
        scn: dq "%s",0
        equal: db "Equal",10 , 0
        notequal: db "Not Equal" , 10, 0

        palin: dq "Palindrome",10 , 0
        nonPalin: dq "Nonpalindrome" , 10, 0
        prr: dq "",10,0



segment .bss
```

```
        string1: resb 100
        string2: resb 100
        value: resq 1
        l: resq 1
        r: resq 1
        string1_len: resq 1
        string2_len: resq 1

segment .text
        global main
        extern scanf
        extern printf

main:
        push RBP

        xor rax ,rax
        mov rdi , scn
        mov rsi , string1
        call scanf


        xor rcx , rcx
        xor rbx , rbx

first_length:
        mov [string1_len] , rcx
        mov al , [string1+rcx]
        cmp al , 0
        je check

        cmp al , 96
        jg loop_con
        xor al , 20H
        mov [string1+rcx] , al
loop_con:
        mov rcx , [string1_len]
        inc rcx
        jmp first_length

check:
        xor rax , rax
        mov [l] , rax
        mov rax , [string1_len]
        dec rax
        mov [r] , rax


checking_loop:
```

```
        mov rcx , [l]
        mov rdx , [r]
        mov al , [string1+rcx]
        mov bl , [string1+rdx]
        cmp al , bl
        jne print_exit
        mov rcx , [l]
        mov rdx , [r]
        inc rcx
        dec rdx
        mov [l] , rcx
        mov [r] , rdx
        cmp rdx , rcx
        jge checking_loop

        xor rax ,rax
        mov rdi , palin
        call printf

        mov rdi , prr
        call printf
        pop RBP
        ret
print_exit:
        xor rax ,rax
        mov rdi , nonPalin
        call printf

        mov rdi, prr
        call printf
        pop RBP
        ret
```

Task-4

```
segment .data
        scn: dq "%s",0
        equal: db "Equal",10 , 0
        notequal: db "Not Equal" , 10, 0


segment .bss
        string1: resb 100
        string2: resb 100
        value: resq 1
        string1_len: resq 1
        string2_len: resq 1

segment .text
        global main
```

```
        extern scanf
        extern printf

main:
        push RBP

        xor rax ,rax
        mov rdi , scn
        mov rsi , string1
        call scanf

        xor rax , rax
        mov rdi , scn
        mov rsi , string2
        call scanf

        xor rcx , rcx
        xor rbx , rbx

first_length:
        mov [string1_len] , rcx
        mov al , [string1+rcx]
        cmp al , 0
        je second_length
        mov rcx , [string1_len]
        inc rcx
        jmp first_length
second_length:
        mov [string2_len] , rbx
        mov al , [string2+rbx]
        cmp al , 0
        je equality
        mov rbx , [string2_len]
        inc rbx
        jmp second_length

equality:
        xor rax , rax
        mov rbx , [string1_len]
        mov rcx , [string2_len]
        cmp rbx , rcx
        jne print_notequal

        xor rcx , rcx

checking_loop:
        mov rcx , [value]
        mov al , [string1+rcx]
        mov bl , [string2+rcx]
```

```
        cmp al , bl
        jne print_notequal
        mov rcx, [value]
        inc rcx
        mov [value] , rcx
        cmp [string1_len] , rcx
        jne checking_loop

        xor rax, rax
        mov rdi , equal
        call printf
        jmp exit

print_notequal:
        xor rax, rax
        mov rdi , notequal
        call printf

exit:
        pop RBP
        ret
```

Task-5:

```
segment .data
        n db 0
        fmt_in : dq "%lld",10,0
        fmt_out : dq "*",10,0
        fmt_outl : dq "* ",10,0
        fmt_outb : dq " ",10,0
        fmt_outn : db "",10,0
        fmt_out1 : dq "%lld",10,0

section .bss
        str: resb 50

segment .text
        global main
        extern printf
        extern scanf

main:
        push RBP
        mov RAX,0
        mov RDI,fmt_in
        mov RSI,n
        call scanf

        mov RCX,[n]
        dec RCX
```

```
first:
        push RCX

space:
        push RCX
        mov RDI,fmt_outb
        mov RAX,0
        call printf
        pop RCX
        LOOP space

        pop RCX
        push RCX

        mov RAX,[n]
        sub RAX,RCX
        mov RCX,RAX

line:
        push RCX
        mov RDI,fmt_outl
        mov RAX,0
        call printf
        pop RCX
        LOOP line

        mov RDI,fmt_outn
        mov RAX,0
        call printf

        pop RCX
        LOOP first

        mov RCX,[n]
        Add RCX,RCX
        dec RCX

last:
        push RCX
        mov RDI,fmt_out
        mov RAX,0
        call printf
        pop RCX
        LOOP last

        pop RBP
        ret
```