Hafsa Rashid
23K-0064
BAI-4A

COAL

Assignment #3

Q1

```
Include Irvine32.inc

.data
dividend WORD 0D4A4h
divisor BYTE 0Ah
msg BYTE "Recursive Division
 complete ", 0

.code
main PROC
    movzx eax, dividend
    movzx ebx, divisor
    call RecDivide
    mov edx, OFFSET msg
    call WriteString
    call crlf
    exit
main ENDP

RecDivide PROC
    cmp eax, 5h
    jle done
    push ebx
    xor edx, edx
    div ebx
    call WriteDec
    call crlf
    call RecDivide
    pop ebx

done:
    ret
RecDivide ENDP

END main
```

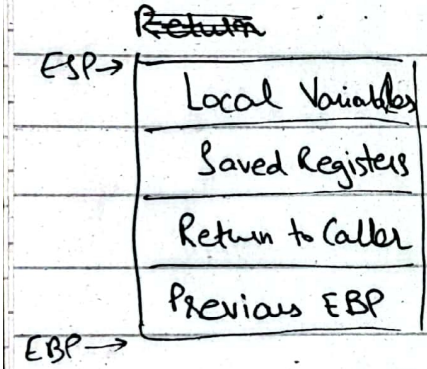
Q2

```
Include Irvine32.inc

.data
Array DWORD 11, 12, 13,
14, 15, 16, 17, 18, 19, 20

ArrSize = ($ - Arr) / TYPE Arr

prompt BYTE "Enter value: ", 0
foundmsg BYTE "Value found
at index: ", 0
notfoundmsg BYTE "Value
not found in the array.", 0

.code
main PROC
mov edx, OFFSET prompt
call WriteString
call ReadInt
mov ebx, eax
mov esi, OFFSET Arr
mov ecx, OFFSE ArrSize
mov edx, 0
call RecSearch
cmp eax, -1
je not_found
mov edx, OFFSET foundmsg
call WriteString
call WriteDec
jmp exit_program

not_found:
mov edx, OFFSE notfoundmsg
call WriteString

exit_program:
call ~~exit~~ calf
exit

main ENDP

~~RecSea~~
RecSearch PROC
cmp edx, ecx
jge not_found
mov eax, [esi + edx * 4]
cmp eax, ebx
je found
inc edx
call RecSearch
ret
```

```
found:
    mov  eax, -1
    ret
RecSearch  ENDP
END main
```

Stack Diagram

- Initial Call:



- First Recursive Call

Local Variables	←ESP
EBX	EBP+20
EDX	EBP+16
ECX	←EBP+12
ESI	←EBP+8
Return to Main	←EBP+4
Previous EBP	←EBP

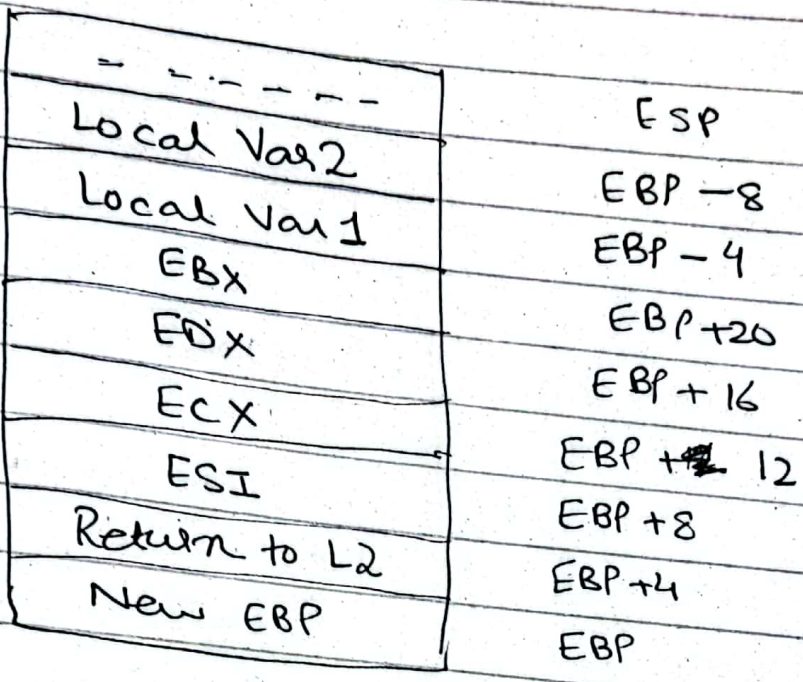
- Second Recursive Call

----------	ESP
Local temp	EBP-4
EBX	EBP+20
EDX	EBP+16
ECX	EBP+12
ESI	EBP+8
Return to L1	EBP+4
New EBP	EBP

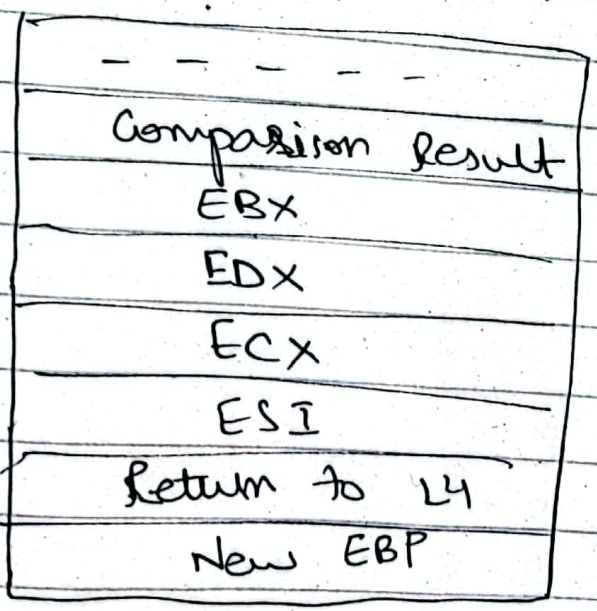
- Third Recursive Call

Local Var2
Local Var1
EBX
EDX

- Third
- Recursive call



- Value found at index 4 (base case)



Q3

```
Include Irvine32.inc

.data
Source BYTE "This is the
 source string", 0
~~target BYTE SIZEOF Source
DUP(0)~~
uniqueChars BYTE 256 DUP(0)

target BYTE SIZEOF Source DUP(0)

.code
main PROC

mov esi, OFFSET Source
mov edi, OFFSET target
mov ebx, OFFSET uniqueChars

CopyLoop:
  mov al, [esi]
  cmp al, 0
  je Done
  mov ecx, al
  cmp BYTE PTR [ebx + ecx], 0
  jne SkipChar
  mov BYTE PTR [ebx + ecx], 1
  mov [edi], al
  inc edi

SkipChar:
  inc esi
  jmp CopyLoop

Done:
  mov BYTE PTR [edi], 0

mov edx, OFFSE Source
call WriteString
call ~~clrf~~ crlf
mov edx, OFFSE target
call WriteString
call ~~clrf~~ crlf

exit
main ENDP
 END main.
```

Q4

```
Include Irvine32.inc

.data
inputString BYTE 256 DUP(0)

prompt BYTE "Enter a
string: ", 0

vowelCount BYTE "Vowel
Count", 0

aCount BYTE "a or A = ",0
eCount BYTE "e or E = ",0
iCount BYTE "i or I = ",0
oCount BYTE "o or O = ",0
uCount BYTE "u or U = ",0

counts DWORD 5 DUP(0)

.code
main PROC
mov edx, OFFSET prompt
call WriteString
mov edx, call

mov edx, OFFSET inputString
mov ecx, SIZEOF inputString
call ReadString
mov esi, OFFSET inputString

ProcessChar:
mov al, [esi]
cmp al, 0
je Display
or al, 20h
cmp al, 'a'
je IncA
cmp al, 'i'
cmp al je IncE
cmp al, 'o'
je IncO
cmp al, 'e'
je IncE
cmp al, 'u'
je IncU
jmp NextChar

IncA:
  jmp counts[0*4]
  jmp NextChar

IncE:
  inc counts[01*4]
  jmp NextChar
```

```
IncI:
    inc counts [2 * 4]
    jmp NextChar

IncO:
    inc counts [3 * 4]
    jmp NextChar

IncU:
    inc counts [4 * 4]

NextChar:
        inc esi
        jmp ProcessChar

Display:
mov edx, OFFSET vowelCount
call WriteString
call dtf crlf
mov edx, OFFSET aCount
call WriteString
mov eax, counts [0 * 4]
call WriteDec
call dtf crlf

mov edx, OFFSET eCount
call WriteString
mov eax, counts [1 * 4]
call WriteDec
call dtf crlf

mov edx, OFFSET iCount
call WriteString
mov eax, counts [2 * 4]
call WriteDec
call dtf crlf

mov edx, OFFSET oCount
call WriteString
mov eax, counts [3 * 4]
call WriteDec
Call dtf crlf

mov edx, OFFSET uCount
call WriteString
mov eax, counts [4 * 4]
call WriteDec
call crlf

exit
main ENDP
END main
```