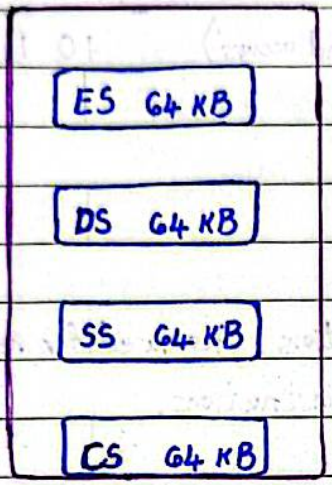
Rayyan Aamir
24K-0687
BCS-3F

Computer Organization & Assembly Language Date:

Assignment #01

Q1 :-

(a)



FFFFFh (Top Memory) : 1 MB memory space

Physical Address (Linear):
(Segment x 10h) + Offset

00000h (Bottom Memory)

Physical Address = (Segment x 10h) + Offset

∴ The segment value is shifted left by 4 bits when multiplied by 10h (hex).
8086 uses 20 bits Linear address in hex.

(b)

(i) 1234 : 5678

Physical Address = (1234 x 10h) + 5678 h
= 12340 h + 5678 h
= 179B8 h (no wrap-around)

```
  12340 h
+  5678 h
---------
  179B8 h
```

(ii) FFFF : 000F

Physical Address = (FFFF x 10h) + 000F h
= FFFF0 h + 000F h
= FFFFF h (no wrap-around)

```
  FFFF0 h
+  000F h
---------
  FFFFF h
```

(iii) F000 : FFF0

Physical Address = (F000 x 10h) + FFF0
= F0000 h + FFF0 h
= FFFF0 h (no wrap-around)

```
  F0000 h
+  FFF0 h
---------
  FFFF0 h
```

Rayyan Aamir
24K-0687
BCS-3F

Date:

(iv) F800 : 9000

Physical Address = (F800 x 10h) + 9000h
= F8000h + 9000h
= 101000h (wrap-around occurs)
= 101000h - 100000h
= 01000h

```
  F8000h
+ 9000h
--------
 101000h
```

Q2:- General Purpose:

EAX:- Used for arithmetic and logic operations. Also used for return values from functions and used for control instructions.

eg:
```
MOV EAX, 10
ADD EAX, 20
```

EBX:- Used as a base pointer for the purpose of memory addressing.

eg:
```
MOV EBX, OFFSET arr
MOV EAX, [EBX]
```

ECX:- Used as a counter in Loops

eg:
```
MOV ECX, 4
loop_example:
        ADD EAX, 5
        LOOP loop_example
```

Index/Base/Stack instruction:

EDX:- Used for multiplication, division and input/output operations.

eg:
```
MOV EDX, 10
IMUL EDX, 5
```

ESI:- It points the source in string and memory operations.

eg:-
```
MOV ESI, OFFSET src
```

Rayyan Aamir
24K-0687
BCS-3F

Date:

EDI:- It points to the destination in string and memory operations.

eg: `MOV EDI, OFFSET dest`

ESP:- It points to the top of the stack; can be used for push/pop function calls.

eg:-
```
PUSH EAX
POP EBX
```

EBP:- It is used to reference function parameters and local variables on stack.

eg:-
```
PUSH EBP
MOV EBP, ESP
MOV EAX, [EBP+8]
```

EIP:- It holds the address of the next instruction to be executed. CPU is responsible for updating it automatically after every instruction.

eg:- `JMP myLabel`

Q3:- 0x1234ABCD starting address = 4000h

Little Endian:

4000h → CD
4001h → AB
4002h → 34
4003h → 12

Big Endian:

4000h → 12
4001h → 34
4002h → AB
4003h → CD

Q4:-

(i) `MOV AX, 8F7AH`

∴ 8F7Ah → 1000 1111 0111 1010 b

Hence;

AX = 8F7AH (H means in hex)

CF = 0

SF = 0

Rayyan Aamir
24K-0687
BCS-3F

Date:

(ii) ADD Ax, 7AF8H

∴ 8F7A h
+ 7AF8 h
10A72 h

Hence;
AX = 0A72 h
CF = 1
SF = 0

(iii) MOV BX, 0FA77 H

∴ FA77 h ⟶ 1111 1010 0111 0111 b

Hence;
BX = FA77 H
CF = 1
SF = 0

(iv) INC BX

∴ FA77 H + 1 = FA88 H

Hence;
Bx = FA88 H
CF = 1
SF = 1

Q5:- (i), (ii) & (iii)

```
INCLUDE Irvine32.inc
Seconds EQU 60
Minutes EQU 60
Hours EQU 24
Days EQU 7
.data
    finalresult DWORD ?

.code
main PROC
```

Rayyan Aamir
24K-0687
BCS-3F

Date:

```
mov eax, Seconds
imul eax, Minutes
imul eax, Hours
imul eax, Days

mov finalresult, eax        ; saving result into memory

mov eax, finalresult
call WriteDec               ; printing the result using eax
call CRLF
exit
main ENDP
END main
```

Q6:-

```
INCLUDE Irvine32.inc
.data
    myArray WORD 1,2,3,4,5,6,7,8,9,10
    size EQU 10
    temp WORD ?

.code
main PROC
    mov bx, [myArray]
    mov esi, OFFSET myArray
    mov ax, myArray
    mov temp, ax
    mov ecx, size-1
    mov esi, OFFSET myArray
reverseArr:
    mov ax, [esi+2]
    mov esi, ax
    add esi, 2
    loop reverseArr
```

Rayyan Aamir
24K-0687
BCS-3F

Date:

```
mov ax, temp
mov esi, ax
mov ecx, size
mov esi, OFFSET myArray
printArr:
    mov ax, esi
    movzx eax, ax
    call WriteInt
    call CRLF
    addesi, 2
    Loop printArr
    exit
main ENDP
END main
```

Q7 :-

(a) For Little endian :-

3000h	3001h	3002h	3003h
D4	C3	B2	A1

(b) For big endian, at lowest address, the Most Significant Byte will be stored and at the highest address, the Least Significant Byte will be stored. (Opposite of Little Endian)

3000h	3001h	3002h	3003h
A1	B2	C3	D4

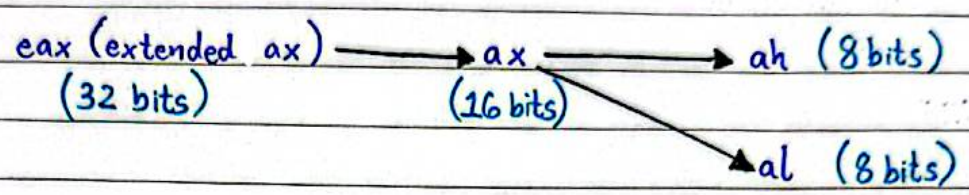
(c) mov eax, [3000h]

Rayyan Aamir
24K-0687
BCS-3F

Date:

Q8:-

The general purpose registers in x86 are 32-bit by default. They can be accessed in smaller chunks for better efficiency and flexibility when storing 16 bits or 8 bits. The x86 architect is compatible with older 16 bit and 8 bit processer ; hence the registers can be split as shown:



This design allows program written for older CPUs be still executed which enhances flexibility.

For example, when storing a hex. character code for 'C' which is 43h. Using eax (32 bits), the processor will allocate 4 bytes for it even though 43h is 1 byte, which makes operations slower when working with strings or characters.
However, using ah or al (both 8-bits), the processor allocates 1 byte which is the same byte size needed to store 43h (1byte); hence not wasting any memory unnecessarily.

Q9:-

```
INCLUDE Irvine32.inc
.data
    varA  DWORD  10
    varB  DWORD  50
    varC  DWORD  5
    varD  DWORD  15

.code
main PROC
    mov eax, varA
    add eax, varB
```

Rayyan Aamir
24K-0687
BCS-3F

Date:

```
mov ebx, var C
add. ebx, varD

sub eax, ebx
mov varA, eax      ; moving final result back to varA
call WriteDec
exit
main ENDP
END main
```

Q10:-

① One to one relationship is when a single entity is ~~only~~ related to only one other entity.

Eg:- A single assembly language instruction corresponds to only a single machine code instruction.

One to many relationship is when a single entity is related to multiple identities.

Eg:- A single high-level language instruction is equal to multiple lines/instructions in assembly and machine code.

② Registers are present inside the CPU, hence instant access. Whereas, for memory access, first address is placed on the address bus then assert. the processor's read pin. Then it takes one clock cycle for memory chip to respond. Finally, data is copied from data bus to destination operand.

③ Machine independent language refers to such a language which can be compiled and executed on any computer system unless the program uses a user-defined libraries ; eg:- C++, Java, Python.
Machine dependent language are depended on the type of processor and cannot be compiled/assembled on every computer system.
eg:- assembly language.

Rayyan Aamir
24K-0687
BCS-3F

Date: _____

④ Computer Architecture is the way hardware components operate and connected together to form computer system. Computer Organization is concerned with the structure and behaviour of computer system as seen by the user.

⑤ Carry Flag (will be set to 1)

⑥ The source file contains the code written by programmer while the Listing file contains the source code, Line number, numeric addresses of instructions and machine code bytes of each instructions.

⑦ Registers are a part of CPU hence ~~memory~~ accessing data is very fast, whereas accessing through memory requires going through buses and possibly memory hierarchy.

⑧ Real Address Mode has 1 MB of memory from 00000 h to FFFFF h. Program can access any part of memory. MS-DOS runs in real address mode. However, in protected mode, each program has total 4 GB of memory capacity available. The Operating System assign memory to each running program. Programs are prevented from each other's memory. Modern OS requires memory protection & multi-tasking, which Real Address Mode Lacks.

⑨ Address Bus, Data Bus and Control Bus are essential components. The address bus hold the address of the instruction and data when the currently executing instruction transfers data between CPU and memory. The data bus transfers data and instructions between CPU and memory. The control bus uses binary signals to synchronize actions of all devices connected to the system.