Sample Questions for Exam Prep

Q. How much megabytes of memory can be accessed using 30 bits of addressing.

_____ MB.

Q. What will be the content of memory (in HEX) after the execution of the following code?

```
[org 0x0100]
jmp start
num1: db 0xA
      dw 0x1234
      dd 0xAB05EF09
start: mov ax, [num1+5]
       add ax, [num1+2]
       mov [num1], ax
       mov ax, 0x4c00
       int 0x21
```

Memory Contents after execution:

Address	Num1+0	+1	+2	+3	+4	+5	+6
Content	17	B4	12	09	EF	05	AB

Q. Give the value of the zero flag, carry flag, sign flag, and the overflow flag after each of the following instructions if AX is initialized with 0x1254 and BX is initialized with 0x0FFF. Assume these instructions run one sequentially.

a. add ax, 0xEDAB

b. add ax, bx

c. add bx, 0xF001

d. add bl, 0x2C

Q. Identify whether instruction given below are valid or invalid.

Instructions	Valid/invalid
`mov [bp-si], 2`	invalid
`mov [bl], 2`	invalid
`mov word [bx], 2`	valid
`mov [bp+bx+si], 2`	invalid
`mov word [bx+ax], 3`	invalid

Q. Given the following contents of registers and memory, what would be the value of ax register after executing the instruction given below. Show your workings.

`mov ax, [bp+si+15]`

`CS: 0x1E0A, DS: 0x1EED, SS: 0xFFEF, BP: 0x011F, SI: 0x0114, DI: 0x0112, BX: 0x0115`

Physical Memory Addresses	Memory Content
0x00132	0900
0x00138	0700
0x1E2E2	0500
0x1F112	0A00
0x1F118	0600

AX = 0700

Q. State whether the jumps in the following assembly codes will be taken or not. Show complete working.

(a)

```
mov cl, 0xFA
mov al, 0x72
sub al, cl
jb exit
```

(b)

```
mov ax, 0x7FBE
mov bx, 0xFFEC
add ax, bx
jo exit
```

(c)

```
mov ax, 0xF153
mov bx, 0x7251
xor ax, 0x195B
cmp ax, bx
jl exit
```

Q. Given a number in the **ax** register, write an assembly code that clears 0th, 2nd and 9th bit in ax, and sets 8th, 12th and 15th bit in ax. Your code should only contain 2 instructions. Code with more than 2 instructions will not gain any marks. The numbering of the bits starts from the right side as shown below.

15	14											3	2	1	0

Q. Write a code to check if a number *num1* is divisible by 4. You are only allowed to use shifting and logical instructions. If the number is divisible by 4, store 1 in the variable divBy4.

```
mov ax. [num1]
shr ax, 1
jc exit
shr ax, 1
jc exit
mov byte[divBy4], 1
exit: ; terminate
```

Alternate solution

```
test [num1], 3
jnz exit
mov byte[divBy4], 1
```

Q. For the code snippet given below, write the byte sized data value that is stored in memory label lab2 after the execution of the program. Briefly explain the working of this program.

```
[org 0x100]
        jmp start

lab1:   db -3, -4, -1, -2
lab2:   db 0

start:  mov bx, 1
        mov dl, [lab1]

loop1:  cmp dl, [bx+lab1]
        jae C1
        mov dl, [bx+lab1]

C1:     add bx, 1
        cmp bx, 4
        jne loop1

        mov [lab2], dl

        mov ax, 0x4c00
        int 0x21
```

Negative numbers are stored in an array, but we are using the jump instruction **jae**, which is valid for the unsigned numbers. This program finds out the maximum unsigned number that is stored in the memory array lab1. In unsigned format, -1 has the maximum value (i.e. 0xFF). Therefore, -1 is stored in memory label lab2, after the execution of the program.

Q. Given the following sequential set of instructions of same program, write down the values of CF, PF, ZF after each instruction (initially all flags are zero).

	CF	PF	ZF
`xor ah, ah`	0	1	1
`mov al, 0x4A`	0	1	1
`shl al, 2`	1	1	0
`rcr ah, 3`	0	1	0
`sub ah, al`	1	0	0

Q. Formula to convert Celsius to Fahrenheit is °F = 1.8°C + 32°. Write a subroutine that it takes Celsius value as parameter, calculate Fahrenheit value in local parameter and return Fahrenheit temperature value in Ax Register. You have to take care of stack structure properly

Note: To multiply by 1.8, first multiply by 18 and then divide by 10, ignoring the remainder.

Q. You have to write a program that **inverts the background colors** of the display. Assume some areas of the screen have a black background while others have white. Write a program which converts the black background areas to white and vice versa.

The screen is currently displaying characters in many different foreground colors and some of them are blinking as well. You are not allowed to modify the foreground colors or blinking attributes

```
[org 0x100]
mov ax, 0xB800
mov es,ax
mov di,0
next:
xor word [es:di], 0x7000
add di,2
cmp di, 4000
jne next
mov ax, 0x4C00
int 21h
```

Q. Write a code that counts the number of blank spaces in the display memory. The count should be saved in the count variable. Use string instructions only. No credit will be given if string instructions are not used. Assume that the screen has black background with white foreground.

```
[org 0x100]
jmp start
count dw 0
start:
mov ax, 0xB800
mov es, ax
mov si, 0
mov cx, 2000
mov bx, 0
loop1: lodsw
cmp ax, 0x0720
jne  loop2
add bx, 1
loop2: loop loop1
mov [count], bx
mov ax, 0x4C00
int 21h
```

Q. Which interrupt will be hooked after execution of following code?

```
[org 0x100]
jmp start
;;;; myISR is written here

start:
xor ax, ax
mov es, ax
mov [es: 0x110 ], myISR
mov [es : 0x112], CS
mov ax, 4c00h
int 21h
```

int 44h

Q. What is the total size (in bytes) of the interrupt vector table?

256 x 4 = 1024 bytes

Q. The service 4Ch (given in ah) of int 21h terminates a program and releases memory. Write a code which disables the service 4Ch of int 21h. All other services of int 21h should work properly.

```
[org 0x100]
jmp start
oldisr: dd 0

my_isr:
cmp ah, 4ch
je end
jmp far [oldisr]
end: iret

start:
xor ax, ax
mov es,ax
mov ax, [es:21h*4]
mov [oldisr], ax
mov ax, [es:21h*4+2]
mov [oldisr+2], ax
cli
mov word [es:21h*4], my_isr
mov [es:21h*4+2], cs
sti
mov dx, start ; now terminate but leave my_isr in memory
add dx, 15
shr dx, 4
mov ax, 0x3100
int 21h
```

Q. Write a piece of code that disables the timer interrupt in the PIC mask register.

```
[org 0x0100]
in al, 0x21
or al, 1
out 0x21, al
mov ax, 0x4c00
int 0x21
```

Q. Consider the following decomposition of the instruction processing pipeline.

Fetch Instruction (FI): Read the next expected instruction into a buffer.

Decode Instruction (DI): Determine the opcode and the operand specifiers.

Fetch Operands (FO): Calculate the effective address of each source operand and fetch each operand from the memory. Operand in registers need not to be fetched.

Execute Instruction (EI): Perform the indicated operation and store the result if any, in the specified destination operand location.

Write Operand (WO): Store the result in memory.

Given below is a set of instructions. There implementation through pipelining has some data hazards. You have to solve those hazards by using stalling method.

I1: mov bx, 0

I2: mov word [n1], ax

I3: add word [n1], bx

I4: add word [n1], 1

I1	FI	DI	FO	EI	WO							
I2		FI	DI	FO	EI	WO						
I3			FI	DI	stall	stall	FO	EI	WO			
I4				FI	stall	stall	DI	stall	stall	FO	EI	WO

Q: It takes 15µs to complete one instruction in a non-pipelined processor. We were able to convert the circuit to a 6 stage pipeline processor. Stage 1 to 6 take 2µs, 1.5µs, 3µs, 4µs, 1.5µs, 3µs resp. Time to move from one pipe stage to another is 2µs. Calculate the following values for pipelined and non-pipelined processor.

- Clock Cycle
- Latency
- Throughput for 46 instructions
- Speedup of pipeline for 75 instructions

Q. Consider the (partial) memory contents given below. Suppose following sequence of addresses is accessed by the CPU.

6, 0, 15, 120, 253, 1, 248, 9, 4, 51, 2, 1

Memory Contents	
Decimal Address	Data
0	2
1	3
2	55
4	9
6	90
7	65
9	5
15	41
51	80
120	100
150	77
170	50
174	32
187	52
248	7
253	2

Index	TAG	DATA
0		
1		
2		
3		
4		
5		
6		
7		

Now consider the 8-word cache shown below. Assume that the cache is initially empty. When inserting an element into the cache, if there are multiple empty slots for one index, you should insert the new element into the first available slot.

Use the direct-mapped cache to facilitate memory access for the memory sequence above. You should fill in the binary form of the Tag values. Show the final contents of the cache in the table, and compute the hit rate

Hit Rate: _____