16.317: Microprocessor-Based Systems I

Spring 2012

Exam 1 Solution

1. (20 points, 5 points per part) ***Multiple choice***

For each of the multiple choice questions below, clearly indicate your response by circling or underlining the single choice you think best answers the question.

a. Which of the following is not an element of a processor software model?

i. Register set (what registers are available, what their purposes are)

ii. What operations the processor can perform

iii. Organization of the memory space

iv. Organization of the physical pins used to interface with the outside world

v. Types of data the processor can use

b. Each statement below names and describes a functional unit on the 80386DX. Which of these descriptions are correct?

A. Bus unit: handles interface to memory and devices outside the processor
B. Execution unit: terminates professors who do not get tenure
C. Segmentation and paging unit: handles memory management and protection services
D. Prefetch unit: translates instructions into microcode operations
E. Decode unit: accesses memory to fill queue of instructions waiting to be decoded

i. Only A

ii. A and C

iii. A, C, and D

iv. A, C, D, and E

v. A, B, C, D, and E

1 (cont.)

c. Given AX = 0009H, DX = 0000H, and (DS:0100H) = 0002H, what is the result of the instruction: DIV WORD PTR [0100H]?

i. AX = 0000H, DX = 0009H

ii. AX = 0009H, DX = 0000H

iii. AX = 0004H, DX = 0001H

iv. AX = 0001H, DX = 0004H

v. AX = DEADH, DX = BEEFH

d. Assume the stack is initially empty before the following sequence of instructions:

```
PUSH EAX
PUSH EBX
PUSH ECX
PUSH EDX
PUSH SI
PUSH DI
```

What is the value of the stack pointer after the final PUSH instruction?

i. FFFEH

ii. FFF8H

iii. FFF2H

iv. FFEAH

v. 0000H

2. (40 points) ***Memory accesses and addressing modes***

Each MOV instruction in the table below demonstrates one of the addressing modes of the 80386DX. Complete the table by determining:

- The address being used for the specified memory operand.
- Whether or not that address is aligned
- The actual byte, word, or double word being transferred, organized in the same way it would be in the register. (Please clearly show how many bytes are being transferred.)

Assume the contents of memory and the following registers are as shown below:

EAX: 000000B4H
EBX: 00000006H
ESI: 00000010H
EDI: 00000012H
EBP: 00000008H
ESP: 00000004H
DS: 3FFFH
SS: 4001H

Address		
3FFF0H	00	01
3FFF2H	02	24
3FFF4H	20	12
3FFF6H	FE	ED
3FFF8H	AB	EE
3FFFAH	CA	BA
3FFFCH	EE	FF
3FFFEH	16	31
40000H	72	02
40002H	FE	B6
40004H	19	78
40006H	AA	CC

Address		
40008H	BB	DD
4000AH	11	23
4000CH	58	D1
4000EH	52	2A
40010H	AF	FA
40012H	EA	1D
40014H	AD	AB
40016H	04	C0
40018H	FE	81
4001AH	18	77
4001CH	EC	E0
4001EH	17	76

Instruction	Address	Aligned? (Yes/No)	Actual data transferred to register
MOV EAX, [22H]	*DS:22H = 3FFF0 + 22 =* ***40012H***	***No***	*Double word from address 40012 =* ***ABAD1DEAH***
MOV AX, SS:[BP]	*SS:BP = 40010 + 0008 =* ***40018H***	***Yes***	*Word from address 40018 =* ***81FEH***
MOV AL, [SI+03H]	*DS:SI+03H = 3FFF0 + 0010 + 03 =* ***40003***	***Yes***	*Byte from address 40003 =* ***B6H***
MOV EAX, [BX+SI+05H]	*DS:BX+SI+05H = 3FFF0 + 0006 + 0010 + 05 =* ***4000B***	***No***	*Double word from address 4000B =* ***52D15823***

3. (40 points) ***Assembly language***

For each instruction sequence shown below, list all changed registers and/or memory locations and their new values. Where appropriate, you should also list the state of the carry flag (CF).

a. (13 points) Initial state:

EAX: 00000000H
EBX: 0000000AH
ECX: 00000000H
EDX: 00000000H
CF: 0
ESI: 00000008H
EDI: FFFF0000H
EBP: 00000400H
ESP: 00002000H
DS: 2110H
SS: 1000H

Address		
21100H	04	00
21102H	10	10
21104H	89	01
21106H	20	40
21108H	02	00
2110AH	00	16
2110CH	17	03
2110EH	FF	00
21110H	1E	00
21112H	06	00
21114H	08	00
21116H	0A	00

Instructions:

```
LSS   DI, [BX+SI]
MOVSX AX, [04H]
ADD   AX, 13H
MOV   [DI], AX
```

Solution:

LSS DI, [BX+SI]→ DI = word at address DS:BX+SI
= word at 21100 + 000A + 0008
*= word at 21112 = **0006***
SS = word at address DS:BX+SI+2
*= word at 21114 = **0008***

MOVSX AX, [04H]→ AX = sign-extended byte from address DS:04
= sign-extended byte from 21100 + 04
= sign-extended byte from 21104
*= 89H sign-extended to a word = **FF89H***

*ADD AX, 13H → AX = AX + 13H = FF89H + 13H = **FF9CH***
CF = 0

MOV [DI], AX → memory at address DS:DI = contents of AX
memory at 21100 + 0006 = FF9CH
byte at 21106 = 9CH
byte at 21107 = FFH

3 (cont.)
b. (14 points) Initial state:

EAX: 00000000H
EBX: 00000000H
ECX: 00000000H
EDX: 00000000H
CF: 0
ESI: 00000008H
EDI: FFFF0000H
EBP: 00000400H
ESP: 00002000H
DS: 3000H
SS: 1000H

Address		
30000H	04	00
30002H	10	10
30004H	89	01
30006H	20	40
30008H	04	08
3000AH	00	16
3000CH	17	03
3000EH	FF	00
30010H	1E	00
30012H	00	00
30014H	FF	FF
30016H	FF	FF

Instructions:

```
NEG   BYTE PTR [08H]
NOT   BYTE PTR [09H]
ADD   AL, [08H]
SUB   AH, [09H]
IMUL  AH
```

Solution:

NEG BYTE PTR [08H]→ *byte at DS:08H = -(byte at DS:08H)*
= -(byte at 30008)
= -04H = FB + 1 = ***FCH***

NOT BYTE PTR [09H]→ *byte at DS:09H = ~(byte at DS:09H)*
= ~(byte at 30009)
= ~08H = ***F7H***

ADD AL, [08H]→ *AL* *= AL + byte at DS:08H*
= 00H + FCH = ***FCH***
CF ***= 0***

SUB AH, [09H]→ *AH* *= AH – byte at DS:09H*
= 00H – F7H = -(F7) = 08 + 1 = ***09H***
CF ***= 0***

IMUL AH → *AX* *= AH * AL, using signed multiplication*
*= 09H * FCH = 9 * -4 = -36*
= -(0024H) = FFDB + 1 = ***FFDCH***

c. (13 points) Initial state:

EAX: 0000003CH
EBX: 00000044H
ECX: 00000004H
EDX: 00008181H
CF: 0
ESI: 00000008H
EDI: FFFF0000H
EBP: 00000400H
ESP: 00000040H
DS: 1000H
SS: 8000H

Address		
10000H	11	22
10002H	33	44
10004H	55	66
10006H	77	88
10008H	99	AA
1000AH	BB	CC
1000CH	DD	EE
1000EH	FF	01
10010H	12	23
10012H	34	45
10014H	56	67
10016H	78	89

Instructions:

```
ROL    AX, 12
ADC    DX, 0
SAR    AX, CL
XOR    AX, DX
```

Solution:

ROL AX, 12 → *AX = AX rotated left by 12 bits*
= 003CH rotated left by 12 bits
(bits being rotated are underlined)
*= **C003H***
*CF = last bit rotated out = **1***

ADC DX, 0 → *DX = DX + 0 + CF = 8181H + 1 = **8182H***

SAR AX, CL → *AX = AX >> CL, keeping sign intact*
= C003H >> 04H
(bits shifted out are underlined)
*= **FC00H***
*CF = last bit rotated out = **0***

XOR AX, DX → *AX = AX XOR DX*
= FC00H XOR 8182H
= $1111\ 1100\ 0000\ 0000_2$ XOR
$1000\ 0001\ 1000\ 0010_2$
*= $0111\ 1101\ 1000\ 0010_2$ = **7D82H***