16.317: Microprocessor-Based Systems I

Summer 2012

Exam 1

July 20, 2012

Name: ____ **ID #:** _________________________

For this exam, you may use a calculator and one 8.5” x 11” double-sided page of notes. All other electronic devices (e.g., cellular phones, laptops, PDAs) are prohibited. If you have a cellular phone, please turn it off prior to the start of the exam to avoid distracting other students.

The exam contains 3 questions for a total of 100 points. Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

The last two pages of the exam (beginning with page 7) contain reference material for the exam: a list of the 80386 instructions we have covered thus far. You may detach these pages and do not have to submit them when you turn in your exam.

You will have 1 hour to complete this exam.

Q1: Multiple choice	/ 20
Q2: Memory addressing	/ 40
Q3: Assembly language	/ 40
TOTAL SCORE	/ 100

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 one of the operation types implemented by most microprocessors?

i. Logical operations

ii. Matrix operations

iii. Program control operations

iv. Data transfer operations

v. Arithmetic operations

b. Which of the following statements about data storage are true?

A. Registers provide long-term data storage because of their capacity, while memory provides storage locations that can always be quickly accessed.
B. Memory locations are typically referenced by address; registers are typically referenced by name.
C. In a segmented memory architecture, only a subset of the total address space is accessible at any given time.
D. Data stored in registers is accessed using "immediate addressing," a term that got its name from the fact that registers are so fast they can be accessed "immediately."
E. When a couple is engaged to be married, they often go to stores to set up wedding registers, a process that most grooms-to-be love more than life itself.

i. A and B

ii. B and C

iii. C and D

iv. A, B, and C

v. B, C, and D

1 (cont.)

c. Given AX = 0013H, CX = 0003H, and DX = 0000H, what is the result of the instruction: DIV CX?

i. AX = 0013H, DX = 0000H

ii. AX = 0004H, DX = 0001H

iii. AX = 0001H, DX = 0004H

iv. AX = 0006H, DX = 0001H

v. AX = 0001H, DX = 0006H

d. Given AX = 00FFH, BX = 0008H, and DX = 0000H, what is the result of the instruction IMUL BL?

<u>NOTE:</u> This problem originally contained a typo, with the instruction written as "IMUL BX". In that case, choice (ii) would be the correct answer—that instruction would normally change DX and AX, but DX remains 0000H

i. AX = 00F8H

ii. AX = 07F8H

iii. AX = FFF8H

iv. AX = 00F8H, DX = 00FFH

v. Pigs fly out of every window on the fourth floor of Ball Hall.

2. (40 points) ***Memory addressing***

Assume the state of the 80386DX registers are as follows:

- (CS) = 1100H
- (DS) = 5020H
- (GS) = 4097H
- (IP) = FEDCH
- (ESI) = 70838E97H
- (EDI) = 102030AAH
- (EBX) = FFEE2245H
- (EBP) = 12345EACH

Complete the table below by:

- Calculating the physical address that corresponds to each given logical address.
- Answering the alignment-related question given in the third column for each address.
  - Be sure to justify your answer in each case.

Logical address	Physical address	Alignment-related question
CS:IP		If you access a double word at this address, is the access aligned?
GS:DI+45H		For what data sizes (byte, word, double word) will an access to this address be aligned?
DS:BX+DI		What is the smallest value you could add to this address to make it an aligned address for double word accesses?
GS:BX+SI+ 102H		What is the largest amount of data that can be accessed in an aligned access to this address?

3. (40 points, 20 points per part) ***<u>Assembly language</u>***

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

a. Initial state:

EAX: 00000000H
EBX: 0000000AH
ECX: 0000021EH
EDX: 0000FF00H
CF: 0
ESI: 00000008H
EDI: FFFF0000H
DS: 3170H
ES: 3171H

Address				
31700H	04	00	08	00
31704H	10	10	20	20
31708H	89	01	19	91
3170CH	20	40	60	80
31710H	02	00	AB	0F
31714H	00	16	11	55
31718H	17	03	7C	EE
3171CH	FF	00	42	D2
31720H	86	75	30	90

Instructions:

```
MOVSX EAX, BYTE PTR [SI+03H]

LES   SI, [10H]

SUB   AX, DX

NEG   WORD PTR [BX+12H]

ADD   AL, CL
```

3 (cont.)

b. Initial state:

EAX: 00000000H
EBX: 00001000H
ECX: 00000002H
EDX: 000000F0H
CF: 0
ESI: 00002000H
EDI: FFFF1000H
DS: AC00H

Address				
AE000H	04	00	02	10
AE004H	10	10	15	AA
AE008H	89	01	05	B1
AE00CH	20	40	AC	DC
AE010H	04	08	05	83
AE014H	00	16	02	06
AE018H	17	03	19	78
AE01CH	FF	00	12	24
AE020H	1E	00	20	07

Instructions:

```
MOV   AL, [SI+0BH]
```

```
SAR   AL, CL
```

```
ROL   AL, 5
```

```
AND   AL, DL
```

```
MOV   [DI+BX+1EH], AL
```

The following pages contain references for use during the exam: a table containing the 80386 instructions we have covered thus far. You may detach these sheets from the exam and do not need to submit them when you finish.

Remember that:

- Most instructions can have at most one memory operand.
- Brackets [ ] around a register name, immediate, or combination of the two indicates an effective address. That address is in the data segment unless otherwise specified.
    - Example: MOV AX, [10H] → contents of DS:10H moved to AX
- Parentheses around a logical address mean "the contents of memory at this address".
    - Example: (DS:10H) → the contents of memory at logical address DS:10H

Category	**Instruction**	**Example**	**Meaning**
Data transfer	Move	`MOV AX, BX`	`AX = BX`
	Move & sign-extend	`MOVSX EAX, DL`	`EAX = DL, sign-extended to 32 bits`
	Move and zero-extend	`MOVZX EAX, DL`	`EAX = DL, zero-extended to 32 bits`
	Exchange	`XCHG AX, BX`	`Swap contents of AX, BX`
	Load effective address	`LEA AX, [BX+SI+10H]`	`AX = BX + SI + 10H`

	Load full pointer	`LDS AX, [10H]`
		`LSS EBX, [100H]`
			`AX = (DS:10H)`
			`DS = (DS:12H)`
			`EBX = (DS:100H)`
			`SS = (DS:104H)`

Arithmetic	Add	`ADD AX, BX`	`AX = AX + BX`
	Add with carry	`ADC AX, BX`	`AX = AX + BX + CF`
	Increment	`INC [DI]`	`(DS:DI) = (DS:DI) + 1`
	Subtract	`SUB AX, [10H]`	`AX = AX – (DS:10H)`
	Subtract with borrow	`SBB AX, [10H]`	`AX = AX – (DS:10H) – CF`
	Decrement	`DEC CX`	`CX = CX – 1`
	Negate (2's complement)	`NEG CX`	`CX = -CX`

	Unsigned multiply (all operands are non-negative, regardless of MSB value)	`MUL BH`
		`MUL CX`
		`MUL DWORD PTR [10H]`
			`AX = BH * AL`
			`(DX,AX) = CX * AX`
			`(EDX,EAX) = (DS:10H) * EAX`

	Signed multiply (all operands are signed integers in 2's complement form)	`IMUL BH`
		`IMUL CX`
		`IMUL DWORD PTR[10H]`
			`AX = BH * AL`
			`(DX,AX) = CX * AX`
			`(EDX,EAX) = (DS:10H) * EAX`

	Unsigned divide	`DIV BH`
		`DIV CX`
		`DIV EBX`
			`AL = AX / BH (quotient)`
			`AH = AX % BH (remainder)`
			`AX = EAX / CX (quotient)`
			`DX = EAX % CX (remainder)`
			`EAX = (EDX,EAX) / EBX (Q)`
			`EDX = (EDX,EAX) % EBX (R)`

Category	Instruction	Example	Meaning
Logical	Logical AND	AND AX, BX	AX = AX & BX
	Logical inclusive OR	OR AX, BX	AX = AX | BX
	Logical exclusive OR	XOR AX, BX	AX = AX ^ BX
	Logical NOT (1’s complement)	NOT AX	AX = ~AX

Shift/rotate (NOTE: for all instructions except RCL/RCR, CF = last bit shifted out)	Shift left	SHL AX, 7
		SAL AX, CX
			AX = AX << 7
			AX = AX << CX

	Logical shift right (treat value as unsigned, shift in 0s)	SHR AX, 7	AX = AX >> 7 (upper 7 bits = 0)
	Arithmetic shift right (treat value as signed; maintain sign)	SAR AX, 7	AX = AX >> 7 (upper 7 bits = MSB of original value)
	Rotate left	ROL AX, 7	AX = AX rotated left by 7 (lower 7 bits of AX = upper 7 bits of original value)
	Rotate right	ROR AX, 7	AX=AX rotated right by 7 (upper 7 bits of AX = lower 7 bits of original value)
	Rotate left through carry	RCL AX, 7	(CF,AX) rotated left by 7 (Treat CF & AX as 17-bit value with CF as MSB)
	Rotate right through carry	RCR AX, 7	(AX,CF) rotated right by 7 (Treat CF & AX as 17-b8t value with CF as LSB)