EECS 2110

Professor
Brent Nowlin

Computer Architecture and Organization

Midterm Opportunity EECS 2110

Computer Architecture and Organization ---- Spring 2021

Printed name____

Academic Honesty Statement: I received no outside assistance in the completion of this opportunity, and all of the work included in this submission is mine and solely mine. By submitting this opportunity, I state that I have not used any unauthorized materials, including, but not limited to, opportunities from previous semesters, evaluation materials produced by others (including persons not enrolled in this course), and the Internet in the completion of this opportunity. I state that the only materials I used in the completion of this opportunity are my notes and textbook. I also state that I did not supply any assistance to other students, nor did I receive any assistance. I have not uploaded any material on this assignment to any website, nor have I photocopied any portion of it with the intent of disseminating it to any persons other than my instructor. I understand that if I do violate academic honesty guidelines, I will be penalized in the grading of the exam and/or my overall course grade, up to and including a course grade of F due to academic dishonesty, which may lead to expulsion from the University. Lastly, I also understand that, if I do cheat and my cheating isn't noted upon grading this opportunity, I will be haunted by the memory of my transgression for the rest of my miserable life.

Signed ______________________________ Date _________

Instructions: Complete the two-part problem, showing **ALL** work. No credit will be given for answers having no work to back them up.

Again, no calculators are allowed, nor computers, nor cell phones, nor smart watches…..no text messaging…..trust me, your cell phone will be fine without you for a couple of hours...and so will you.

Show all work for all questions in this opportunity to show what you know. I can't give any credit without the work being shown.

EECS 2110

Professor
Brent Nowlin

Computer Architecture
and Organization

EECS 2110 – Midterm Opportunity I: Spring 2021

Name ________________________

1. The four general-purpose registers are the AX, BX, CX, and DX. There are functions that are frequently associated with each of these registers, and an often-used term for each of these registers. Match the registers with their alternate terms and functions from the list below. Only use the number of the function; you don't have to repeat what is written below. (6)

AX: also known as the ____________________________, functions:

BX: also known as the ____________________________ , functions:

CX: also known as the ____________________________ , functions:

DX: also known as the _____________________________ , functions:

Alternate terms:

Base register
Counter register
Data register
Accumulator

Functions:

1. Frequently used to specify offsets into segments: used for array indexing and pointing into strings
2. Frequently used as a register to hold the result of arithmetic operations
3. Frequently used to count through loop iterations and other repeated program instructions, e.g. for-next loops
4. Used as a good general-purpose register; sometimes involved in arithmetic operations such as MUL and DIV

2. The operation of removing the newest item on the stack is called_____________

   and the operation of adding a new item to the stack is called ____________. (3)

3. The EAX register
   a) Is 32-bit
   b) is used with many assembly language instructions
   c) can be used as one of its component registers AL, AX, or AH
   d) All of the above

4. The flags register has several flags of interest. Write the flag for each description. (9)

___________ Setting this flag will result in step-by-step execution of the program
___________ Setting this flag will decrement the index register on string move operations
___________ This flag reflects the status of the preceding operation’s sign bit (MSB)
___________ Setting this flag will enable interrupts
___________ This flag is set if the preceding operation had an overflow
___________ This flag is used by the CPU for BCD operations
___________ This flag is set if the preceding operation had a carry out
___________ This flag is set if the result of the preceding operation is zero
___________ This flag reflects the parity of the result of the previous operation

5. Some uses of the stack are
    a. Storing data
    b. Storing register contents for later retrieval
    c. Storing addresses
    d. All of the above
    e. None of the above

6. In the event of an interrupt,
    f. the CS is pushed onto the stack
    g. the IP is pushed onto the stack
    h. the flags register is pushed onto the stack
    i. all of the above

7. Given the segment values, what are the physical addresses that are affected by the following code, as well as the values in those locations after the code executes? (5)

Seg reg contents:
ES = 02200h
DS = 02FF0h
SS = 09087h
CS = 0A000h

Code:
```
mov     BX, 2040h
mov     [BX], SS
```

Memory Address (hex)	Data (hex)

The statements below are false. Fix one or two words to make it true. (2 pts each)

8. The ADD EAX, EBX instruction places the sum in the EBX register

9. Registers are POPped off the stack in the same order they are PUSHed

10. The MOV instruction affects the sign and zero flags

11. When defining a variable in the data segment, the assembler (1) links the name to the variable's offset, (2) reserves space in the data segment, (3) specifies the data type for instruction use, and (4) pushes its value onto the stack

Fill in the blank / circle the right answers…. (1-2 pts per blank/circle)

12. The Intel x86 processors use word alignment in the form of (big-endian / little-endian ***circle one***) . This implies that the word's location is given by the location of the (low-order byte / high-order byte ***circle one***). If the word is properly aligned, the word's low-order byte is stored at an (even / odd ***circle one***) byte location, while the high-order byte is stored at the next address, which is (even / odd ***circle one***) .

13. The _____________ register is commonly used for loop counting. There is a special instruction to check for this register being 0, the ___________ instruction. This special instruction implies that a loop should (increment from 0 / decrement to 0 ***circle one***) for efficiency purposes.

14. Using given values of AX, CX, and the CF, fill in the table for the given operation (instruction). Give the resulting AX value and the CF where appropriate. (4ea).

AX: 1011 0101 0011 1110$_2$ CF = 0
CX: 0010 1010 0000 0110$_2$

	Resulting AX		
Operation	Binary value	Hex value	Resulting CF
AND			
OR			
XOR			
NOT (AX)			
SHL			
SAR			

15. What is the…. (1.5-2 pts each)

a. hexadecimal range… of ASCII uppercase letters (A - Z)? ___________________
of ASCII lowercase letters (a – z) ? ____________________
of ACSII numeric symbols (0 – 9)? ____________________

b. x86 assembly instruction … to make character in BL uppercase? ____________
to make a character in BL lowercase?____________
to toggle the case of character in BL? ___________

c. best assembly language instruction to convert the numeric symbol in BL to its numeric value?

16. For the instructions and initial register contents (which are all independent of one another), give the result (noting the destination register) and the flag values after execution. Enter "?" is the flag value is unknown after the operation, and enter "no change" if an instruction doesn't affect a certain flag. Write "unchanged" if an instruction leaves registers unchanged. If an instruction won't assemble, state the reason why across the row (instead of the result and flag values). I started it for you. (2-2.5 ea)

AX = 0101h	CX = 4402h	OF: overflow flag	CF: carry flag
BX = ????	DX = FFEFh	ZF: zero flag	SF: sign flag

	Result	OF	SF	ZF	CF
AND AX, CX	AX = 0002h	0	0	1	0
ADD AX, CX					
DEC AX					
ADD sum, cat					
MOV AL, DX					
INC DX	DX=0FFF0h	0	1	0	0
MOV AX, 65535h					
CMP AX, CX					
MOV BX, AX					
MUL AX, BX					
DIV CX					
MUL AX, BX					
MUL DX					
ADD AX, FFEFh					
OR AX, DX					
TEST AX, CX					
XOR AX, DX					
AND CX,03FH					

17. There are several tasks associated with the definition of a variable. Circle all of the tasks that apply. (4 pts)

    a. The assembler uses the name to equate the variable with a value
    b. The assembler creates a link between the name and its offset in the data segment
    c. The processor uses the variable's name in all of the instructions using the variable
    d. The assembler creates a stack equal to the value given in the stack directive
    e. The variable's initial value is pushed onto the stack
    f. The variable's value is initially set to the value as given in the name definition
    g. The assembler reserves space in the data segment to store the variable

18. The MUL and DIV instructions are a little bit different that the other arithmetic instructions. (8 pts)

    a. What are the valid operands for these instructions?

    b. What is the destination operand(s) of the MUL instruction for the given source operand sizes?

    Byte: ________________________________

    Word: _________________________________

    Double word: ______________________________

    c. What is the destination operand(s) of the DIV instruction for the given source operand sizes?

    Byte: ________________________________

    Word: _________________________________

    Double word: ______________________________

(12)

19. Ima Loserstudent, takes an assembly language class. For one of his assignments, he found this piece of code on laying around on internet, and he chooses to submit it for his project instead of writing the project code for himself. Ima didn't bother to check the code and neither did his buddies he shared it with. The code is supposed to divide the unsigned number (passed into a procedure via AX) by 64_{10} and add 42_{10} to the division's remainder (returned in AX), and restore any registers other than AX to their original values.. Fix the code by changing the given instructions and comments, and **optimize it** for execution. (10)

```
; initialization stuff

            calculate proc  ;there is a number in ax. Calculate (AX%64 + 42)

                    mov     DX, 0  ; clear out DX for divide
                    not     DX
                    mov     AX, 64; initialization
                    div     BX
                    add     AX, 52; add 52 to the remainder
                    pop     DX     ; restore DX
                    ret

            calculate endp
```

20. The following code, which is incorrect, is supposed to add the numbers from 1 to 50 decimal, and place them in a variable called summation. Correct the code so that it will function properly. (10 pts)

```
            Summation       db      ?

Sum:        MOV     AX, 0002h
            ADD     AX, summation
            INC     AX
            CMP     AX, 0050h
            JGE     Sum
```

21. Write the code fragment to add the first num_values values contained in word_array (an array of words whose size exceeds num_values). Place the sum of the words in a variable called array_sum. Don’t worry about overflow or checking to make sure num_values doesn’t exceed the array size. I started it for you.(12 pts)

```
…(beginning stuff)

    Array_sum       DW    0
    Num_values      DB    0
    Word_array      DW    0001h, 0002h, 0003h, 0004h, … up to 00FFh

    …(other stuff)

    ; assume the word_array has been changed by the code placed here…

            MOV   CH, 012h; for some reason

            MOV   array_sum, 01234h ; for some other reason

    ; and assume that num_values has been set properly by other code here…

            MOV   BX, offset word_array

            MOV   CL, num_values

            ADD   ____________________________________________




            DEC   CX



    …(rest of program)
```

(12)

22. What is the result after the following code fragment executes? (8 pts)

```
Ht_ft       DB      7
Ht_in       DB      3
; code segment
MOV     BL, 12
MOV     AL, ht_ft
MUL     BL
MOV     DL,  ht_in
MOV     DH, 00h
ADD     AX, DX
….
```

AX = ________________________

BX = ________________________

CX = ________________________

DX = ________________________

23. Consider the following program…..

```
…….
.DATA
New_word DW 1C5Fh
.CODE
Main PROC
        mov ax, si
        mov ax, 0005h
        add ax, new_word
         mov bx, 00ABh
        Main ENDP
END Main
```

Circle all of the problems from the list below that apply to the code.

A. The add instruction will not execute
B. The program will not exit cleanly
C. We did not load the address of the data segment
D. This is a useless program

24. *(note that I didn't write this question, but feel it's somewhat simple and good for a laught. I did put some editorial comments for your reading pleasure…)* For some reason you have been writing assembly for the Russian government (**note:** that's gotta be a great gig), and you're one multiplication away from ensuring that Garfield the Cat will be the next president of the United States (**note:** Garfield would certainly be an upgrade from recent presidents / presidential candidates). However, the mighty Vladimir Putin himself has coded this multiply (**note:** how did Putin ever learn assembly language???), and to alter it would be treason (**note:** time for a job change). Instead you must recalculate valid ranges for all other code to fit these 4 lines (**note:** that's a clever way of staying alive…). For what values of K will Vlad's code cooperate? Assume unsigned multiplication and that the code will assemble. (4 pts)

```
MOV BX, 42h
MOV AX, K
MUL BX
MOV Multiplication_Result, AX
```

25. Consider the following assembly language instructions, with the data definitions (and locations) given. Fill in the blanks for the code snippet (only give information for the destination operands). Assume word_array is located at offset 0020h (15 pts)

```
.DATA
word_array    DW    000EH  0010H  0012H  0014H
        ; ….code segment
MOV  BX,    offset word_array ;    ______________ will contain ________________
MOV  DX,    word_array ;           ______________ will contain ___________________
MOV  SI,    BX;                    ______________ will contain ___________________
INC SI;                     ______________ will contain ___________________
MOV  AX,    [SI] ;          ______________ will contain ___________________
INC SI;                     ______________ will contain ___________________
MOV CX, [SI];               ______________ will contain ___________________
```

26. Assume the data segment contains the following data, given as ASCII values (hint – this is a character type of code fragment, not arithmetic)…offsets from the start of the data segment are given. What does the following code do, i.e. output? (8 pts)

```
Data segment:
Offset 0010h:   !  r  n  e  u  t  f  s  f  a  o  o  s  T  d  s
Offset 0020h:   a  i  o  t  l  a  =  c  s  p  g  m  n  u  i  l
Offset 0030h:   r  p  t  s  s  `  y  h  l  a  l  o  i  o  s  N
…
                … ( code and other stuff…)
                MOV   SI, 003Eh
                MOV   CX, 0017h
Output_loop:    mov   al, [SI]
                Call writestring       ;outputs character in the AL register to the screen
                                       ; and moves the cursor position 1 space to the right
                DEC   CX
                DEC   SI
                DEC   SI
                JCXZ  Done
                JMP   Output_loop
                … (rest of code)
```

(23)

29. This is a fun problem. Complete the missing code portions below to implement a nested FOR loop. Don't worry about error checking, wrong values, overflow, etc. I generated the coding template already – just fill in the missing pieces. (8 pts)

```
Outer_loop_value      DW     20; this is the number of times the outer loop will execute
Inner_loop_value      DW     10; this is the number of times the inner loop will execute
; beginning code segment stuff
                      MOV CX, outer_loop_value
Outer_loop:
                      PUSH  _______________
                      MOV   CX, inner_loop_value
Inside_loop:
                      ; do the inside loop stuff here, then…
                      DEC   ______________
                      JCXZ  ______________
                      JMP   ______________
Inside_loop_done:
                      POP   CX
                      DEC   ______________
                      JCXZ  outer_loop_done
                      JMP   _______________
Outer_loop_done:
                      ; rest of code segment
```

27. Write the assembly language fragment to perform the given procedure. Assume Output_number, Input_number, A, B, and N are defined in the data segment as words. Don't worry about overflow or other problems that may give erroneous output. (11 pts)

Output_number = A * Input_number^N + B