Name: DT

CSC231 Midterm Exam
Fall 2012

This is a closed-book, closed-notes in-class exam given under the rules of the honor code. You cannot discuss any details or exchange information with anybody except the instructor. You have 50 minutes to answer 6 of the questions. If you answer more than 6 answers, the top 6 answers will be counted.

Problem #1

- Write a program that computes and stores in an array of 16-bit words the first 10 powers of 2. The array starts with 0 in all 10 cells then stores 1 in the first cell, 2 in the second, 4 in the third, etc.
- Is a 16-bit format sufficient to store all the numbers? yes! $2^{10} < 2^{16-1}$ which is the largest unsigned that can be stored in a 16-bit format.

```
Power:  dw   0,0,0,0,0
        dw   0,0,0,0,0
        :

        mov  ax, 1
        mov  ecx, 10
        mov  ebx, Power
for:    mov  word [ebx], ax
        add  ax, ax
        add  ebx, 2
        loop for
```

Name:_____

Problem #2

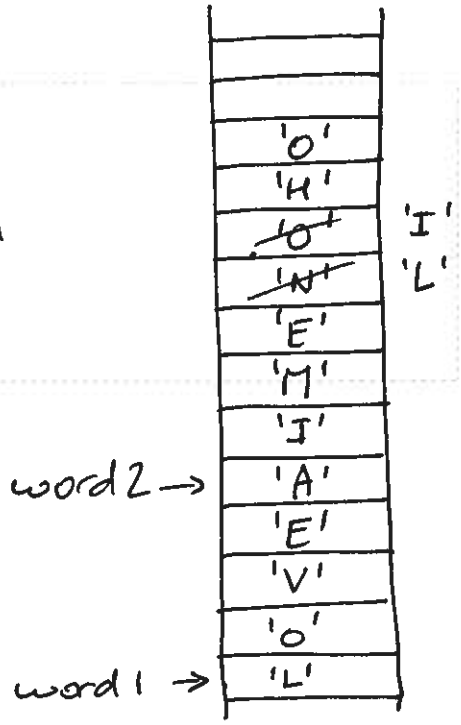
- What is in word3 once the three instructions below have finished executed?

```
word1   db   "LOVE"
word2   db   "AIME"
word3   db   "NOHO"
        mov  ah,0   ;assume 0 in ah
1       mov  al, byte [word1]
2       add  ah, byte [word2+1]
3       mov  [word3], ax
```

① al ← 'L'

② ah ← 'I'

③ word3 ← "LIHO"



Problem #3

1) • Assume we decide to use words of 10 bits to represent 2's complement numbers. What would be the *range* of integers we could store in such a format? Explain your answer.

2) • Could 1024 be stored in such a format?

3) • What is the binary representation of -1 in this format?

1) -2^9 to 2^9-1 = -512 to 511

2) No, 1024 cannot be represented in 2's complement or as an unsigned number in a 10-bit format

3) -1 = 11 1111 1111 in binary in a 10-bit format

Name:_____

Problem #4

- How many times does the loop below repeat?

```
            mov     ecx, 0
for:        inc     eax
            dec     ebx
            loop    for
```

the loop will go 2^{32} times

- What numbers are printed by the loop below?

```
            mov     ecx, 10
            mov     eax, 0
for2:       call    print_int
            mov     eax, ecx
            dec     ecx
            loop    for2
```

0 10 8 6 4

Name:_____

Problem #5

- What values are left in eax, ebx, ecx, and edx at the end of these instructions?

```
        mov     eax, 0
        mov     ebx, eax
        sub     ecx, ecx
        and     edx, 0x00000000
        add     al, 1
        or      al, 0xf0
        loop    for
for:    mov     bx, dx
        or      ebx, 0xffff0000
```

eax : 00 00 00 F1
ebx : FF FF 0000
ecx : FF FF FFFF
edx : 0000 0000

Problem #6

1) - Perform the addition (in binary or hex, by hand) of -2 and 1024 coded as 16-bit 2's complement numbers. Is the result correct as a 16-bit 2's complement number?

2) - What decimal number is represented by 0xFFFFEFFF in a 32-bit 2's complement format? (Note that there is one *E* in the pattern of *F*s!)

1) -2 : 1111 1111 1111 1110
1024 : 0000 0100 0000 0000

carry flag → (1) 0000 0011 1111 1110

0 3 F E

= 1022 decimal

correct answer

2) flipping all the bits and adding 1 we get

0x0000 1001

$= 1 \times 16^3 + 1$
$= 4096 + 1$
$= 4097$

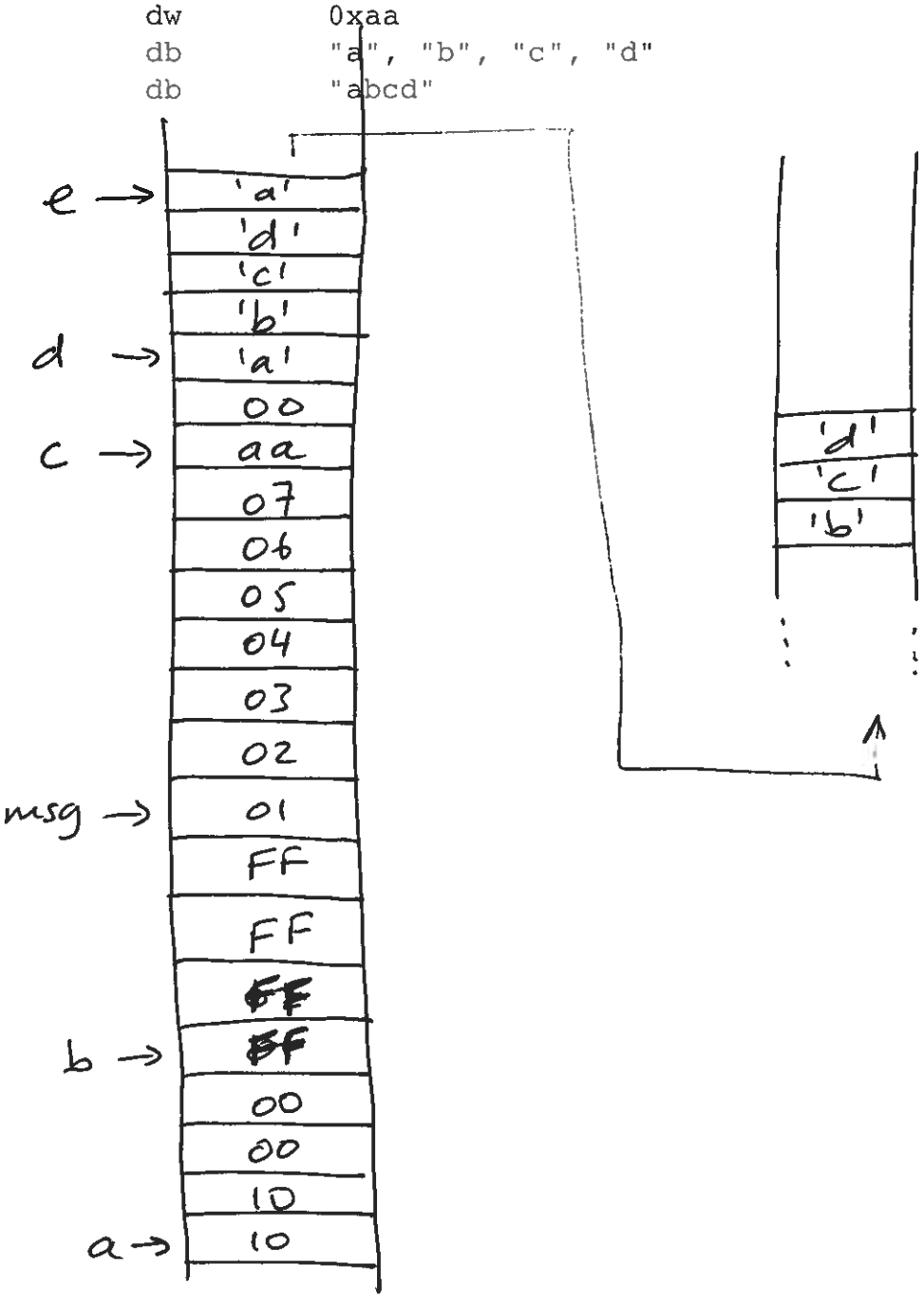
⇒ the number was −4097

Name:_____

Problem #7

Show the contents of the memory with whatever format is appropriate once the variables shown below have been loaded in it:

```
            section .data
a             dd        0x1010
b             dd        -1
msg           db        1,2,3,4,5,6,7
c             dw        0xaa
d             db        "a", "b", "c", "d"
e             db        "abcd"
```



Name:_____

Problem #8

Look at the code below. Is the **mov** instruction valid?

Will it create a segmentation fault? If not, what value will get stored in eax?

What *addressing modes* are used for the 2 operands of the **mov** instruction?

```
            section     .data
msg1        db          "midterm exam"
msg2        db          "fall 2012"

            section     .text
            global      asm_main

asm_main:
          mov         eax, msg1-msg2
```

Assume msg1 = 0, then msg2 = 12. They are both label and represent addresses. Addresses are 32-bit numbers.

So msg1 - msg2 = 0 - 12 = -12

The instruction is equivalent to

mov eax, -12