68/100 +17

Name: _Giancarlos Marte_____

University of Massachusetts - Boston **Dr. Ronald Cheung**
Computer Architecture and Organization **CS 341 - Spring 2021**

In-Class, Open Book Mid-Term Examination
March 23, 2021

The work on this examination is to be your own and you are expected to adhere to the UMass-Boston honor system. All questions can be answered by one or two short sentences. Do not try to make up for a lack of understanding by providing a rambling answer.

Note: I give partial credit! Show all work!

1. (20 points) Short Questions

a. (2 points) What is the use of the Control Bus?
It is used to transfer information between the processor, memory and I/O devices.

OK

b. (2 points) If %eax = 0x100200, what is wrong with the following Intel assembly instruction?

cmpl 8(%eax), 0x100100
8(%eax) is not possible because 0x100200 is a constant number, not a used memory address. You would have to do cmpl %eax, 0x100100.

-2: wrong

c. (4 points) Name an advantage and a disadvantage of a processor that has more bits in its address bus and data bus over another processor that has fewer bits on these busses?

Advantage: The processor will be faster by being able to transfer more data at a time._____OK_____

Disadvantage: More bits in those busses will increase the processor price and physical size._____OK_____

d. (2 points) What is the difference between dynamic RAM and static RAM: dynamic RAM is only retained for a short amount of time whereas static RAM is retained as long as the power is on. Dynamic RAM is also slower and cheaper than static RAM. OK

e. (6 points) For the following program:
void main( ){ char * ptr; *ptr = 5;}

i) Which part of the memory can you find ptr?
You can find ptr in the RAM stack.

OK

ii) What does the program do when you run it on the linux server?
It declares a char pointer. Then it assigns that pointer to the integer 5. The only issue is that you have to cast the integer 5 into a char if not an error will occur during compiling.

-2: wrong

iii) What does the program do when you run it on the tutor VM?
If you run it in the tutor VM you will not get any error as there is no operating system to catch the issue.

OK

f. (4 points) In Lab 2, why do you use the function pgm_read_byte(&a[i]) to read the EEPROM content instead of just using a[i]? You use the function to read the EEPROM content because Arduino does not let you use a[i] to access it.

48134402

2. (20 points) Evaluations

a. (10 points) The content of 16 memory locations starting at 0x00100250 is:

```
00100250        01 23 45 67 58 02 10 00
                89 ab cd ef 00 01 10 11
```

Show the content of %ecx, %eax after executing these instructions:

```
movl   0x00100254, %ecx
movl   -4(%ecx), %eax
```

%ecx= _23___ -5: wrong

%eax = _01___ -5: wrong

b. (10 points) Show the hex value of the eax register and state of the specified condition flags after executing the instruction.

```
movl    $0xf0f0f0f0, %eax
movl    $0, %ecx
xorl    %ecx   %eax
```

%eax = 0x _f0f0f0f0___ OK

CF = __0__ OK

SF = __1__ OK

ZF = __1__ -2: wrong

OF = __0__ OK

3. (20 points) Stack Operations

You are debugging a program on the tutor VM. The content of 16 memory locations starting at 0x003fffe0 is:

```
003fffe0   bc f0 10 00 00 bd 89 ab
           00 00 e8 01 00 00 00 cc
```

%eax = 0x12345678
%esp = 0x003fffe8

What are the memory and register content after you execute the instruction: pushl %eax ?

0x003fffe0: _00cc_____
0x003fffe1: _0000_____
0x003fffe2: _e801_____ -8: wrong. Each address stores 8 bits
0x003fffe3: __0000_____

0x003fffe4: _89ab_____
0x003fffe5: __00bd_____ -8: wrong. Each address stores 8 bits
0x003fffe6: __1000_____
0x003fffe7: ___bcf0_____

%eax = _0x12345678___ OK

%esp = __0x003fffe4___ OK

4. (40 points) Assembly language program

Write a C callable assembly language function (change_case.s) to change the alphabets in a string to either upper case or lower case. The user will enter an option. The change_case function will return the string with the correct case or it will return the error string if an invalid option is entered.

The function prototype of the assembly language function in C is shown as:

extern char * change_case(char option);

Your assembly language function should get the option from a C main function shown below:

```
/* change_casec.c: C driver for the changing case
   function. Users can enter an option:

     option = 'L' or 0x4c: change all to lower case

   The function will return the error string if an
   invalid option is selected.
*/

#include <stdio.h>
extern char * change_case(char option);

int main()
{
 char option;
 char *ptr;
 printf("Enter case change option:  ");
 scanf("%c", &option);

 ptr = change_case(option);
 printf("\nThe new string is : %s\n", ptr);
 return 0;
}
```

 # Change case assembly language program
 # You only have to implement the option = 'L'
 # Return the error string if option != 'L'
 # You can find the ascii code chart at the end
#

```
        .data
list:  .asciz "All Good Things Come To Those
Who Wait"
error: .asciz "Invalid option"

opt: .asciz "L"
converter: .long 32

          .text
          .globl change_case
change_case:
      # stack frame
      pushl %ebp
      movl %esp, %ebp

      # get option memory start
      movl 8(%ebp), %ecx

      # get the letter
      movb (%ecx), %dl  <-- no need ind address

      # check if option invalid
      cmpb %dl, opt
      jz exit

      # move string into eax
      movl list, %eax   <-- need $list

tolower:
      # get the first letter of string
      movb (%eax), %dl
      # check if null
      cmpb $0, %dl
      jz exit
      # check if already lower case
      movl %dl, %ecx
      addl $20, %ecx
      cmpl $81, %ecx
      je convert
      incl %eax
      jmp tolower

convert:
      # convert to lower
      movl %dl, %ecx
      addl $20, %ecx
      movl %ecx, %eax
      # loop
      incl %eax
      jmp tolower

exit:    <-- need to return pointer
```

```
        movl %ebp, %esp
        popl %ebp
        ret
        .end
```

-0: get input argument OK
-0: check option OK
-1: get data from $list, and ind addr OK
-0: check terminating char OK
-0: change case logic OK
-5: miss returning pointer
-0: stack operation OK
-6

```
.end
```

ASCII Code Chart:

Here is the **ASCII Encoding**, a correspondence of keyboard characters with integers from 0 to 127 (0x7F in hexadecimal, 0177 in octal)

char	hex	oct
NUL	00	000
SOH	01	001
STX	02	002
ETX	03	003
EOT	04	004
ENQ	05	005
ACK	06	006
BEL	07	007
BS	08	010
HT	09	011
NL/LF	0A	012
VT	0B	013
NP/FF	0C	014
CR	0D	015
SO	0E	016
SI	0F	017
DLE	10	020
DC1	11	021
DC2	12	022
DC3	13	023
DC4	14	024
NAK	15	025
SYN	16	026
ETB	17	027
CAN	18	030
EM	19	031
SUB	1A	032
ESC	1B	033
FS	1C	034
GS	1D	035
RS	1E	036
VS	1F	037

char	hex	oct
SP	20	040
!	21	041
"	22	042
#	23	043
$	24	044
%	25	045
&	26	046
'	27	047
(	28	050
)	29	051
*	2A	052
+	2B	053
,	2C	054
-	2D	055
.	2E	056
/	2F	057
0	30	060
1	31	061
2	32	062
3	33	063
4	34	064
5	35	065
6	36	066
7	37	067
8	38	070
9	39	071
:	3A	072
;	3B	073
<	3C	074
=	3D	075
>	3E	076
?	3F	077

char	hex	oct
@	40	100
A	41	101
B	42	102
C	43	103
D	44	104
E	45	105
F	46	106
G	47	107
H	48	110
I	49	111
J	4A	112
K	4B	113
L	4C	114
M	4D	115
N	4E	116
O	4F	117
P	50	120
Q	51	121
R	52	122
S	53	123
T	54	124
U	55	125
V	56	126
W	57	127
X	58	130
Y	59	131
Z	5A	132
[	5B	133
\	5C	134
]	5D	135
^	5E	136
_	5F	137

char	hex	oct
`	60	140
a	61	141
b	62	142
c	63	143
d	64	144
e	65	145
f	66	146
g	67	147
h	68	150
i	69	151
j	6A	152
k	6B	153
l	6C	154
m	6D	155
n	6E	156
o	6F	157
p	70	160
q	71	161
r	72	162
s	73	163
t	74	164
u	75	165
v	76	166
w	77	167
x	78	170
y	79	171
z	7A	172
{	7B	173
|	7C	174
}	7D	175
~	7E	176
DEL	7F	177