In-Class, Open Book Examination I

October 17, 2017

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

1. (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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Disadvantage:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (2 points) What is the tutor command to display registers?
2. (4 points) In Arduino EEPROM, how is checksum used and how is it computed?

1. (2 points) Explain what is incorrect about this instruction:

popl $0x3fffc0

1. (2 points) Difference between the overflow and carry flag
2. (2 points) What is the difference between dynamic RAM and static RAM?

1. (4 points) For an Intel processor, what are the memory contents after executing the following?

movl $0x12345678, %eax

movl %eax, 0x100100

100100: \_\_\_\_\_\_

100101: \_\_\_\_\_\_

100102: \_\_\_\_\_\_

100103: \_\_\_\_\_\_

2. (20 points) Evaluations

a. (10 points) Memory at address 0x00100250 contains the following 16 bytes of data:

00100250 54 02 10 00 01 23 45 67

89 ab cd ef 00 01 10 11

Show the contents of %ecx, %eax after

executing these instructions:

movl 0x00100250, %ecx

movl 4(%ecx), %eax

%ecx= \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

%eax = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

b. (5 points) Show the hex value of the eax register and state of the specified condition flags after executing the instruction.

.text

xorl %eax, %eax

%eax = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ CF = \_\_\_\_

SF = \_\_\_\_ ZF = \_\_\_\_ OF = \_\_\_\_

c. (5 points) Write 1 assembly language instruction (not using the xor instruction) that will get the same result as in b.

3. (20 points) Computer Architecture

How come the following cupid program does not get the right result in a Harvard architecture machine:

#code to get the ID of CPU using instruction 0x0fa2

\_cpuid:

movb $0x0f, cpuid1 # patch in the cpuid first byte

movb $0xa2, cpuid2 # patch in the cpuid second byte

movl $0,%eax # input to cpuid for ID value

cpuid1: # hex for cpuid instruction here

nop # 0x0f replaces 0x90

cpuid2:

nop # 0xa2 replaces 0x90

. . .

4. (40 points) Assembly language program

Write a C callable assembly language function

(go\_low.s) to change an array of characters to

lower case.

The function prototype in C is shown as:

extern void go\_low(char \* a);

You write an assembly language function that uses the pointer argument to read an array of chars, convert all upper case chars to lower case and store the result back in the same array. For chars that are not upper case, leave them unchanged.

The C main program that calls the assembly language function is shown as follows:

/\* go\_lowc.c: C driver for the convert function

\*/

#include <stdio.h>

extern void go\_low(char \*a);

int main()

{

char input[]= “LOL: The Sky is Falling”;

printf("Before conversion:%s\n", input);

go\_low(input);

printf("After conversion:%s\n", input);

return 0;

}

Answers:

1. Short Questions
2. Advantage:

Wider address bus can address more memory locations. Wider data bus can read/write multiple byte variables faster.

Disadvantage:

Need more pin-outs on the hardware die making it more expensive to make. Signals closer to each other can have more cross-talk problems.

1. Tutor command rd – register display
2. The checksum is used to check the correctness of the downloaded program.

The checksum is computed by doing an exclusive-or on all bytes except the last one and then performing a 1’s complement on the result. The result is stored in the last location. During the program download, the exclusive-or operation followed by 1’s complement is performed on the target machine and this result is compared to the last location of the program for correctness.

1. The instruction tries to restore the stack into a number which is not allowed.
2. Carry flag is set when the addition/subtraction of 2 numbers causes carry out of /borrow into the most significant bits.

Overflow flag is set when the sum of 2 numbers with the sign bits off/on yields a resulting number with the sign bits on/off.

1. Dynamic and Static RAMS are both volatile memories. However, DRAM requires refreshing, but SRAM does not.
2. Intel processors store data in memory using the little Endian format:

100100: 0x78

100101: 0x56

100102: 0x34

100103: 0x12

1. Evaluations

a. 00100250 54 02 10 00 01 23 45 67

89 ab cd ef 00 01 10 11

%ecx %ecx + 4 %eax (little endian)

The 0x00100254 value is moved from memory location

to %ecx.

The value moved from memory location 4 + 0x00100254

to %eax is 0xefcd ab89

b. %eax = 0 CF=0

SF=0 ZF=1 OF=0

c. The following instruction will get %eax=0:

movl $0, %eax

1. For a Harvard architecture machine, the code and data segments are in different addressing space. In our sample code, the locations for cpuid1 and cpuid2 will not follow the instruction movl $0, %eax.
2. Assembly Language Program go\_low.s

.text

.globl \_go\_low

\_go\_low:

pushl %ebp # save ebp in stack

movl 8(%esp), %eax # get input address

label:

movb (%eax), %dl # get the content

cmpb $'A', %dl # compare with

# letter A

jb label1 # leave if less than

cmpb $'Z', %dl # compare with

# letter Z

ja label1 # leave if greater

addb $0x20, %dl # change the case to

# lower case

label1:

movb %dl, (%eax) # store data back

# into the array

incl %eax # get next character

cmpl $0, (%eax) # is equal to null?

jne label # if not, loop

popl %ebp # restore ebp

ret

.end