**Name: __Giancarlos Marte___**

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

# Take Home Final Examination

**Assigned: May 12, 2021** **Due: 11:59pm Tuesday May 18, 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. (30 points) Short Questions:

a. (3 points) The counter of the PIT is always changing. What hardware mechanism does the PIT provide to ensure that we are reading the right values?

A counters output latch is used to ensure that the right values are being read. The count is stored until it is read or until the counter is reset.

b. (6 points) Name 2 functions you can program the Programmable Interrupt Controller (PIC) to do.

i) You can make it send an interrupt signal.
ii) You can read interrupt signals. This is used to determine priorities of multiple signals (IRQ's).

c. (6 points) Name two differences between L1 and L2 cache?

i) L1 cache is not visible to assembly language due to hidden register implementation.
ii) L2 is usually larger than the L1 cache and uses SRAM memory.

d. (9 points) The output of sequential logic is based on:

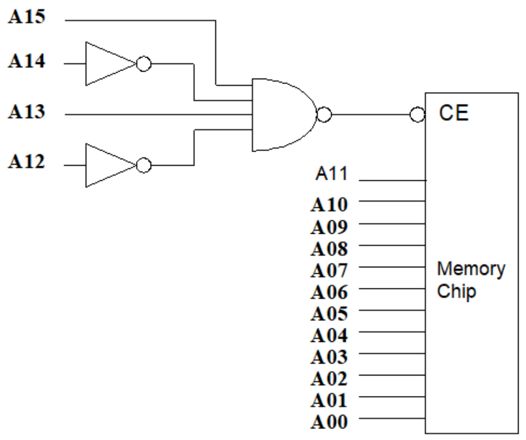
i) present input signals,
ii) the sequence of it's input signals, and
iii) the past outputs.

e. (6 points) Explain how the iret instruction differs from the ret instruction and explain where you would use each of them?

The iret instruction pops %eip, %cs and %eflags. iret is used to exit interrupt procedures and is used when programming the PIC. The ret instruction restores %eip and %esp. It is used to exit procedures and is used in assembly code.

## 2. (20 points) Combinational Logic:

The design of a 16-bit address decoder is shown below:



The memory chip is enabled when CE =1. Use a truth table to determine what addressing bits turn on the chip:
* CE = NOT(NAND(all)) because dot

| A15 | A14 | NOT (A14) | A13 | A12 | NOT (A12) | NAND (all) | CE |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |

What range of memory addresses will be selected by this decoder:

Hex address range: <u>0xA03</u> to <u>0xA15</u>

## 3. (20 points) Timer Programming:

Loading the PIT counter in mp4/mp5 with a value of 30000 will generate an interrupt every 25 msec.

(5 points) what is the counter value to generate an interrupt every 50 msec?
55msec = 65536
55/50 = 1.1
65536/1.1 = 59578.18182
<u>Around 60000</u>

(5 points) what is the counter value to generate an interrupt every 75 msec?
55msec = 65536
55/75 = 0.7333333
65536/0.7333333 = 89367.27273
<u>Around 90000</u>

(10 points) Write a set_timer_count() function below with the count value together with an ISR in C so that the PIT will call an output() function every 75 msec:

```
// 55 * (15/11) = 75
// ISR
void irq0inthandc(void) {
        pic_end_int();
        tickcount++;
        if (tickcount % (15/11) = = 0) {
                output();
        }
}

void set_timer_count(int count) {
        outpt(TIMER_CNTRL_PORT,
TIMER0|TIMER_SET_ALL|TIMER_MODE_RATE
GEN);
        outpt(TIMER0_COUNT_PORT,count&0xff);
/* set LSB here */
        outpt(TIMER0_COUNT_PORT,count>>8); /*
and MSB here */
}
```

## 4. (30 points) Assembly language program

Write a C callable assembly language program (google.s) to find out how many months google's stock (goog) price is above a certain number in 2020.

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

extern int google(int number);

Your assembly language function should get the dollar number from a C main function shown below and return the number of months that the stock price is equal or above the entered number. The program should exit when it reads the stop code 0.

```
------------------------------------------------------
/* googlec.c: C driver for the stock analysis
                    function
*/

#include <stdio.h>
extern int google(int number);

int main()
{
    int n, number;
    printf("Enter the stock price to compare: \n");
    scanf("%d", &number);

    n = google(number);
    printf("Number of months that goog is equal or
over the price of ", %d, "in 2020 is : %d\n", number,
n);
    return 0;
}



------------------------------------------------------
```

```
------------------------------------------------------
# GOOG Stock price analysis assembly program
#  google.s
#
          .globl google
          .data
price:    .long 1260     #11/1/19 price
          .long 1289     #12/1/19 price
          .long 1337     #1/1/20 price
          .long 1434     #2/1/20 price
          .long 1389     #3/1/20 price
          .long 1105     #4/1/20 price
          .long 1320     #5/1/20 price
          .long 1431     #6/1/20 price
          .long 1438     #7/1/20 price
          .long 1482     #8/1/20 price
          .long 1634     #9/1/20 price
          .long 1490     #10/1/20 price
          .long 0        #stop code
          .text
google:


          .end
------------------------------------------------------
```

Instructions:

a) Copy the files you need from /courses/cs341/s21/cheungr/final_exam/.

b) Edit the google.s to include your code. Use the provided make file to build your google.lnx program:

```
make  A=google
```

c) Run and debug your program in the VMs. Create a typescript of the run of the lnx file in the VM. scp it to your cs341/final_exam/ folder at uses.cs.umb.edu for grading.

**Please Note:**

1. Only use data in 2020 for comparison
2. Include the C stack frame in your assembly code.