**EASTERN MEDITERRANEAN UNIVERSITY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CMPE 443**

**Real-Time System Design**

**Midterm Exam**

**2012-2013 Fall Semester**

**November 8, 2012**

**Name-Surname : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Student Number : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Instructor:

Assoc. Prof. Dr. Alexander CHEFRANOV

Time: 110 minutes

MOBILES ARE NOT ALLOWED!!!

YOU CAN BRING ONE A4 SIZED SHEET OF *HAND-WRITTEN* NOTES TO THE EXAM. PHOTOCOPIES ARE NOT ALLOWED AND WILL BE COLLECTED.

READ THE INSTRUCTIONS FOR EACH SECTION CAREFULLY.

**Grading**

|  |  |
| --- | --- |
| **PART I** |  |
| **PART II** |  |
| **TOTAL** |  |

# Part I. Hardware Considerations (50 points)

A. (25 points) Consider 0-address machine instruction set below:

|  |  |
| --- | --- |
| **0-address machine (reverse Polish)** | |
| where the processor has a stack and some supporting hardware, at least a top of stack (TOS) pointer. | |
| **Operation** | **e.g. or comment** |
| load\_literal <int> | effect: TOS:=TOS+1; stack[TOS]:=<int> load a *constant* onto the top of stack; this can be used in arithmetic or to get an address onto the stack for use by a load or a store instruction later (it is splitting hairs to argue whether the literal is an address or a constant which might happen to be used as an address elsewhere) |
| load | effect: stack[TOS]:=memory[stack[TOS]] take the top-of-stack as an address, replace the top-of-stack with the contents of that address. |
| sto | effect: memory[stack[TOS-1]]:=stack[TOS]; TOS:=TOS-2 store contents of top of stack at the address in stack[TOS-1] then pop the value and the address |
| <opcd> | where <opcd> is add | sub |... effect: stack[TOS-1] := stack[TOS-1] <op> stack[TOS]; TOS:=TOS-1 |

For the expression

y=z\*y-(w+z)/(y-w)

1. Give the reverse Polish notation (5 points)

Yzy\*wz+yw-/-=

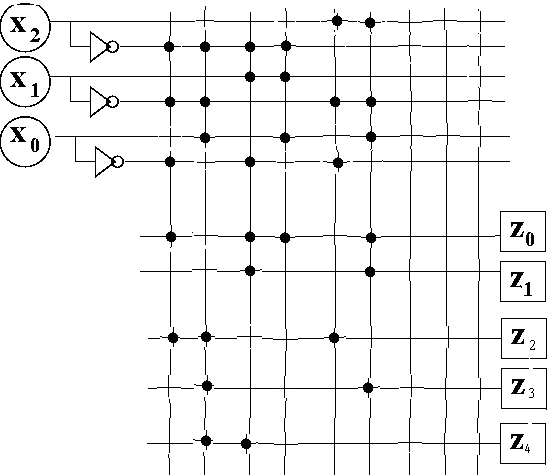
1. Write the code to implement it (10 points)
2. Load\_literal @y
3. Load\_literal @z
4. Load
5. Load\_literal @y
6. Load
7. Mul
8. Load\_literal @w
9. Load
10. Load\_literal @z
11. Load
12. Add
13. Load\_literal @y
14. Load
15. Load\_literal @w
16. Load
17. Sub
18. Div
19. Sub
20. sto
21. Trace the code assuming w=2, x=3, y=3, z=2: show state of the memory cells allocated for the variables and state of the stack initially and after each instruction completion. (10 points)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| After Instruction # | w=2  (addr=100) | Y=3  (addr=101) | Z=2  (addr=102) | Stack  Empty |
|  |  |  |  | 101 |
|  |  |  |  | 101,102 |
|  |  |  |  | 101,2 |
|  |  |  |  | 101,2,101 |
|  |  |  |  | 101,2,3 |
|  |  |  |  | 101,6 |
|  |  |  |  | 101,6,100 |
|  |  |  |  | 101,6,2 |
|  |  |  |  | 101,6,2,102 |
|  |  |  |  | 101,6,2,2 |
|  |  |  |  | 101,6,4 |
|  |  |  |  | 101,6,4,101 |
|  |  |  |  | 101,6,4,3 |
|  |  |  |  | 101,6,4,3,100 |
|  |  |  |  | 101,6,4,3,2 |
|  |  |  |  | 101,6,4,1 |
|  |  |  |  | 101,6,4 |
|  |  |  |  | 101,2 |
|  |  | 2 |  |  |

B. (13 points) Use PLA to create a ROM with 6 memory cells containing the following numbers in the given order: 5,12, 19, 1, 4, 11. For each number, specify its address in your ROM. Each memory cell shall have one and the same number of bits, and this number of bits shall be minimal possible to accommodate the specified six numbers. What is the number of bits of each memory cell? Why? What is the number of address bits? Why? N

Number of bits in the memory cells is 5 because maximal number is 19

Number of address bits is 3 because number of cells is 6.



C. (12 points) Give FPGA implementation of the following computations assuming that FPGA elements can perform any arithmetic/logic operation:

A=(w+x)\*(y-z)

B=(w+x)^2+y

Show necessary operation to be used for each element in your implementation (e.g., element for +, or \*, or ^, etc). Use minimal possible number of elements in your implementation.

# Part II. Software considerations (50 points)

w

x

y

z

IOo

IO

IO

IO

+

\*

-

^

IO

IO

IO

IO

A

2

IO

+

IO

B

IO

IO

IO

IO

IO

IO

1. (15 points) What is the sequence of actions performed by CPU when interruption signal number i is raised?

CPU saves PSW and PC in stack, calculates address of the interruption vector addr=4\*i, fetches from addr an interruption vector and loads it into PC (yields control to an interruption handler)

B. (20 points) Assume a process, Reader, reads data from an external device and writes them into a buffer for keeping N data items. When Reader is ready to read (i.e., buffer is not full), flag read\_ready is raised. Device informs about data availability by raising data\_ready flag. After reading data, Reader resets data\_ready flag, so that device may provide new data. When the buffer is full, Reader resets flag read\_ready. Write C-like pseudocode for Reader. Assume that Reader works infinitely. When the buffer is full, some other process shall empty it and re-raise read\_ready flag. Initially, buffer is empty.

item buffer[N];

item read\_data();

int read\_ready=1, data\_ready=0;

Reader(){

Int i=0;

While(1){

If (read\_ready){

If(data\_ready){

Buffer[i++]=read\_data();//read fro the device&save in buffer

Data\_ready=0;

If(i>=N){i=0; read\_ready=0;}

}//end if data\_ready

}//end if read\_ready

}//end while

}//end reader

1. (15 points) Calculate processor utilization and hyper-period for the following task set:

|  |  |  |
| --- | --- | --- |
| **Task#** | **E** | **P** |
| **1** | **2** | **9** |
| **2** | **2** | **10** |
| **3** | **2** | **6** |
| **4** | **1** | **4** |

U=2/9+2/10+2/6+1/4=(40+36+60+45)/180=181/180

H=9\*4\*5=180