CS 2200 Fall 2014 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

Problem	(Points, min)	Lost	Gained	Running Total	TA
1	1, 0				
2	24, 15				
3	25, 15				
4	15, 10				
5	10, 10				
6	12, 10				
7	13, 15				
Total	100, 75				

- **You may ask for clarification but you are ultimately responsible for the answer you write on the paper.**
- **Illegible answers are wrong answers.**
- **Please do not discuss this test by any means with anyone in the class**
- **Please look through the entire test before starting. WE MEAN IT!!!**

Good luck!

1. (1 point, 0 min) (This is a freebie, you get 1 point regardless) How I prepared for this test (Circle all that apply):
   (a) By paying attention to class lectures
   (b) By going through Kishore's marked up slides
   (c) By reading the book chapters carefully
   (d) By working out past exams
   (e) By working out example problems in the book
   (f) By watching Kishore's video recordings from a previous offering
   (g) By watching Georgia Tech beat Virginia Tech on Saturday
   (h) By playing Angry Birds during class time
   (i) By using class time as nap time
   (j) By just showing up for the test without prep
   (k) By fervently praying that there won't be a test

CS 2200 Fall 2014 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

Processor design

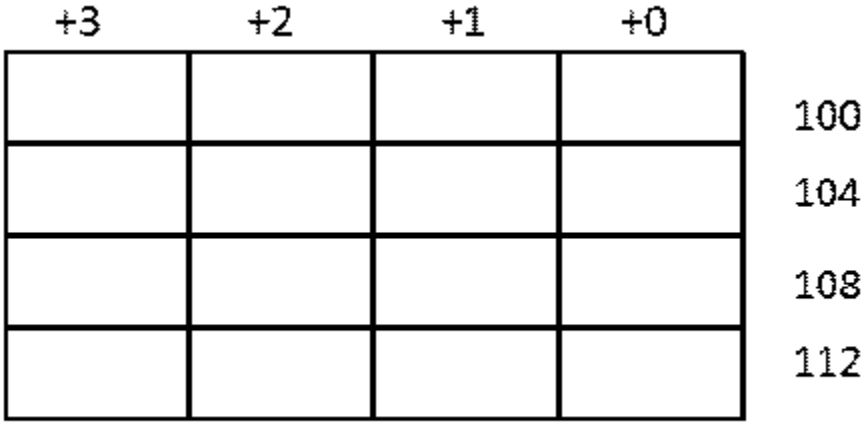
2. (15 min)

(a) (10 points) Given the following data declarations:

```
short a; /* occupies 2 bytes: a_hi a_lo */
short b; /* occupies 2 bytes: b_hi b_lo */
char  c; /* occupies 1 byte */
short d; /* occupies 2 bytes: d_hi d_lo */
```

Consider a **32-bit little-endian** architecture; the architecture supports **L/S** instructions at **byte**, **half-word**, and **word** granularities. How will the compiler lay out above data declarations in memory starting at **memory address 100** to achieve the **best space/time tradeoff**? [Hint: consider packing and alignment].

In the following memory picture each row represents a memory word comprising of 4 bytes, and each cell represents a byte.



MSB **LSB**

(b)

Given the software convention for registers:

```
a0-a2:  parameter passing
s0-s2:  callee saves if need be
t0-t2:  caller saves if need be
v0:     return value
ra:     return address
at:     target address
sp:     stack pointer
fp:     frame pointer
```

Recall that JAL instruction of LC-2200 has the following semantics:

```
JAL at, ra;       ra <- PC_incremented (return address)
          ;       PC <- at (entry point of procedure)
```

The state of the stack is as shown below. To help you out, we have put down the action corresponding to step 1.

(i) (12 points) For each of the other steps, write down:

- What is the action
- Who is responsible for the action

Your answer:

(ii) (2 points) In the above picture, show where the **frame pointer** is pointing to on the stack; where the **stack pointer** is pointing to on the stack.

CS 2200 Fall 2014 Test 1

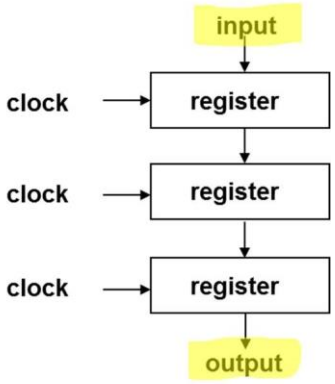
Prism ID:_____

Name:_____ GTID#: 9_____

Datapath and control

3. (15 min)

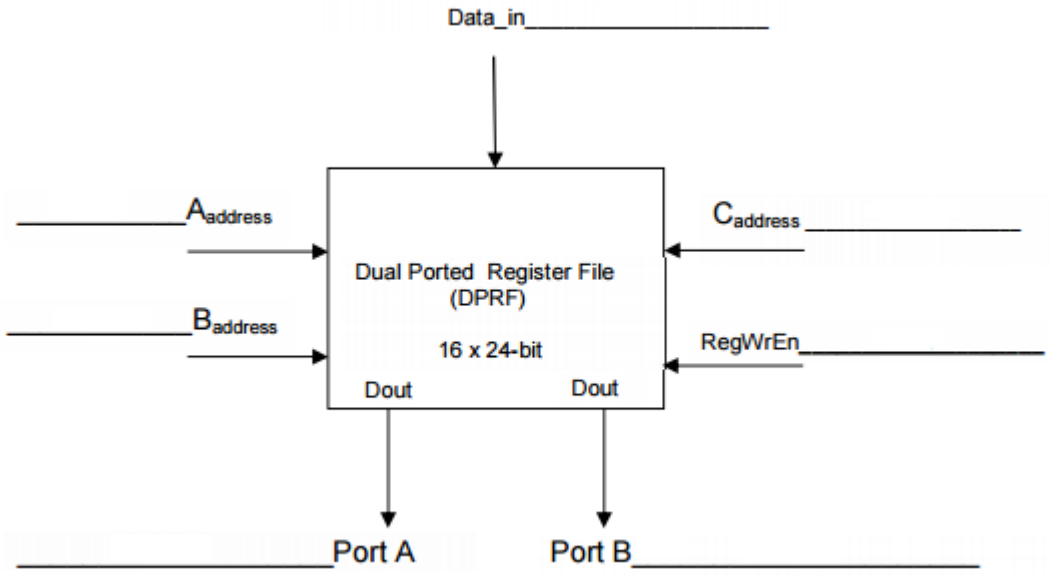
A. (2 points) (circle the correct choice) With positive-edge triggered logic, how many rising clock edges are needed to get the value at the input to the output in the picture below:



1. **One**
2. **Two**
3. **Three**
4. **Four**

B. (7 points) Given the 16-element dual-ported register file (each register is 24-bits), wherein $A_{address}$ and $B_{address}$ are the register addresses for reading the 24-bit register contents on to Ports A and B, respectively. $C_{address}$ is the register address for writing Data_in into a chosen register in the register file. RegWrEn is the write enable control for writing into the register file. Fill in the number of bits needed for the control lines into the register file and the data lines in and out of the register file.

(+1 for each correct answer)

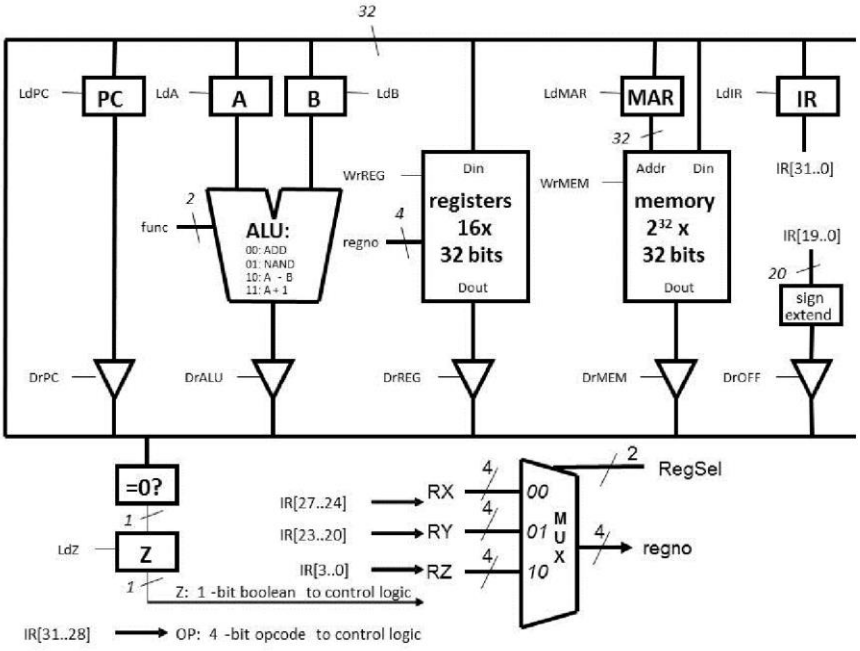


CS 2200 Fall 2014 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

C. All the questions are in relation to the datapath shown below



(a) (2 points) (circle the correct choice)The reason for the "MUX" in the above datapath is…

- to reduce the number of bits in the control ROM for specifying the register number from 4 to 2
- for versatility of the design to allow for future enhancement of the architecture
- to dynamically select the appropriate register depending on the need of a particular microstate
- to ensure that the register number specified is between 0 and 15

(b) (2 points) (circle the correct choice)The duration of a micro state in the FSM-based implementation of LC-2200 is

- One clock cycle
- One macro state
- As many clock cycles as needed by the macro state
- Three clock cycles

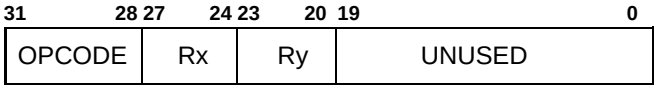
(c) (2 points) (circle the correct choice)The purpose of the "sign extend" box in the above datapath is…

- to convert the 20-bit offset in the IR into an unsigned number
- To convert the 20-bit offset in the IR into a 32-bit 2's complement number
- to convert the 20-bit offset in the IR into a 32-bit 1's complement number
- to reduce the number of wires in the datapath

D.
We have decided to add another addressing mode **autodecrement** to LC-2200. This mode comes in handy for LW/SW instructions. The semantics of this addressing mode with SW instruction is as follows:

```
        SW  Rx, -(Ry) ;       Ry <- Ry – 1
                      ;       MEM[Ry] <- Rx
```

The instruction format is as shown below:



(i) (2 points) State any additional capabilities needed in the datapath to implement this addressing mode.

(ii) (8 points) Assuming such additional capabilities, write the sequence for implementing the SW instruction with this addressing mode (you **ONLY** need to write the sequence for the **execution macro state of the instruction**). For each microstate, show the datapath action (in register transfer format such as A ← Ry) along with the control signals you need to enable for the datapath action (such as DrPC).

CS 2200 Fall 2014 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

Interrupts

4. (10 mins)

A. (8 points)

a) (circle the correct choice) Upon an interrupt before executing the handler the following **DOES NOT** happen implicitly in hardware..

(i) Current PC is saved in $k0
(ii) PC is loaded to point to the interrupt handler code
(iii) The processor registers are saved on the stack
(iv) Interrupts are disabled

b) (circle the correct choice) The interrupt vector table holds the starting addresses of the interrupt handlers for all the known sources of program discontinuities during program execution. This table is

(i) a special piece of hardware in the processor
(ii) implemented in ROM and preset at manufacturing time of the processor
(iii) kept at some designated area of the memory known to the processor and the operating system
(iv) kept in persistent storage such as the hard disk so that it can survive system crashes and power failure

c) (circle the correct choice) The difference between an external device interrupt and an internal trap or exception is that

(i) there is none
(ii) the vector number for a trap/exception is internally generated by the processor
(iii) the vector number for a trap/exception is provided by the user in one of the general-purpose registers
(iv) the vector number that is put out by a given device, changes each time it interrupts

d) (circle the correct choice) Identify which of the following is **NOT** atomic

(i) An individual instruction execution in LC-2200
(ii) Saving all the registers on the stack
(iii) The processor action in the INT macro state
(iv) The processor action in the FETCH macro state

Prism ID:_____

Name:_____ GTID#: 9_____

B. (7 points)
Here is a skeleton for an interrupt handler code with several things missing.

```
save processor registers;
execute device code;
restore processor registers;
return to original program;
```

Fix the above handler so that it will work correctly respecting the requirements of a correct interrupt handler.

For each line you add/replace in the above handler program, place a comment next to it to explain why.

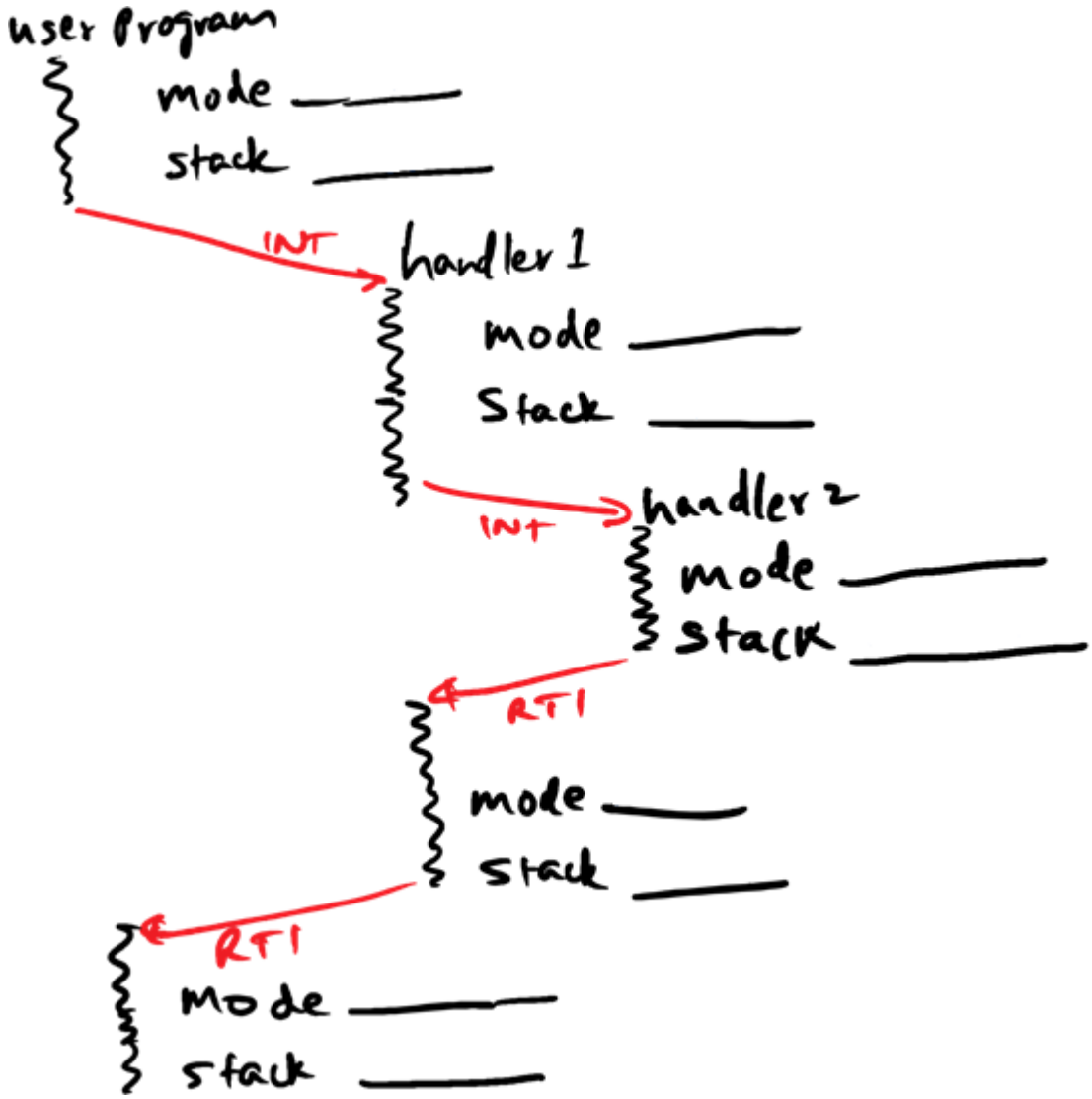
Your answer here:

CS 2200 Fall 2014 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

5. (10 points, 5 mins)Fill in the blanks below to indicate what is the mode (**user, kernel**) of the processor and what is the stack in use (**user, system**)



CS 2200 Fall 2014 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

Performance

6. (15 mins)

a) (10 points)

An architecture has three types of instructions that have the following CPI:

Type	CPI
A	2
B	5
C	1

An architect determines that he can reduce the CPI for B to 3, with no change to the CPIs of the other two instruction types, but with an increase in the clock cycle time of the processor. What is the maximum permissible increase in clock cycle time that will make this architectural change still worthwhile? Assume that all the workloads that execute on this processor use 30% of A, 10% of B, and 60% of C types of instructions.

b) (2 points) (circle the correct choice) Dynamic instruction frequency…

(i) Refers to the type of instructions in the instruction-set

(ii) Refers to the frequency of occurrence of instructions in compiled code

(iii) Refers to the frequency of occurrence of instructions that actually get executed

(iv) Refers to clock frequency of the processor

(v) Is the basis for datapath design

CS 2200 Fall 2014 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

Pipelining

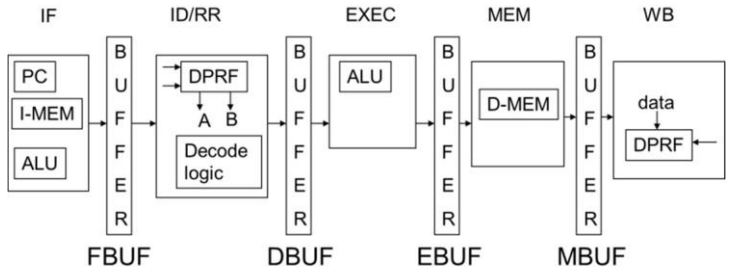
7. (15 mins)

a) (2 points) A highway has 4 lanes between point A and point B. One car enters each of the four lanes every minute at point A. All the cars exit at point B. The time to travel between point A and point B is 20 minutes. In the **steady state** what is the **throughput** observed at point B?

b) (2 points) (circle the correct choice) Comparing the instruction pipeline to Bill's sandwich pipeline, the buffer between the stages of the instruction pipeline serves the same function as

(i) Partially assembled sandwich
(ii) Order form passed between workers
(iii) Combo of the above two
(iv) No comparison

c) (9 points) For the LC-2200 instruction set we are considering a pipelined processor design using a 5-stage pipeline as shown below



Design the DBUF pipeline register for the LC-2200. Do not attempt to optimize the design by overloading the different fields of this register. Consider the needs of each of the four types of instructions in LC-2200: R-type, Itype, J-type, O-type in coming with a complete design of the DBUF (see next page for LC-2200 ISA summary).

CS 2200 Fall 2014 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

LC-2200 Instruction set

- R-type instructions (add, nand):

bits 31-28: opcode
bits 27-24: reg X; dst bits 23-20: reg Y; src
bits 19-4: unused (should be all 0s); bits 3-0: reg Z; src

- I-type instructions (addi, lw, sw, beq):

bits 31-28: opcode; bits 27-24: reg X; dst
bits 23-20: reg Y; base bits 19-0: Imm. Offset

- J-type instructions (jalr):

bits 31-28: opcode; bits 27-24: reg X; target
bits 23-20: reg Y; link bits 19-0: unused

- O-type instructions (halt):

bits 31-28: opcode; bits 27-0: unused

Metrics cheat sheet:

Name	Notation	Units	Comment
Memory footprint	-	Bytes	Total space occupied by the program in memory
Execution time	$(\sum CPI_j) * \text{clock cycle time}$, where $1 \le j \le n$	Seconds	Running time of the program that executes *n* instructions
Arithmetic mean	$(E_1+E_2+\ldots+E_p)/p$	Seconds	Average of execution times of constituent *p* benchmark programs
Weighted Arithmetic mean	$(f_1*E_1+f_2*E_2+\ldots+f_p*E_p)$	Seconds	Weighted average of execution times of constituent *p* benchmark programs
Geometric mean	p^{th} root $(E_1*E_2*\ldots*E_p)$	Seconds	p^{th} root of the product of execution times of *p* programs that constitute the benchmark
Harmonic mean	$1/(\ ((1/E_1)+(1/E_2)+\ldots+(1/E_p))/p\ )$	Seconds	Arithmetic mean of the reciprocals of the execution times of the constituent *p* benchmark programs
Static instruction frequency		%	Occurrence of instruction *i* in compiled code
Dynamic instruction frequency		%	Occurrence of instruction *i* in executed code
Speedup (M_A over M_B)	E_B/E_A	Number	Speedup of Machine A over B
Speedup (improvement)	E_{Before}/E_{After}	Number	Speedup After improvement
Improvement in Exec time	$(E_{old}-E_{new})/E_{old}$	Number	New Vs. old
Amdahl's law	$Time_{after} = Time_{unaffected} + Time_{affected}/x$	Seconds	*x* is amount of improvement