# CS 2200 Spring 2010 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

| Problem | Points | Lost | Gained | Running Total | TA |
|---|---|---|---|---|---|
| 1 | 1 | | | | |
| 2 | 9 | | | | |
| 3 | 10 | | | | |
| 4 | 10 | | | | |
| 5 | 10 | | | | |
| 6 | 20 | | | | |
| 7 | 15 | | | | |
| 8 | 10 | | | | |
| 9 | 15 | | | | |
| Total | 100 | | | | |

- **You may ask for clarification but you are ultimately responsible for the answer you write on the paper.**
- **Illegible answers are wrong answers.**
- **Please look through the entire test before starting. WE MEAN IT!!!**

**Good luck!**

1. (1 point, 1 min) (select one)

**Slam dunk** refers to

- a pagan ritual practiced in Polynesian islands _____
- the act of pouring Gatorade on a coach after winning Superbowl _____
- a weird contest in which grown men leap into the air and try to put a basketball into a hoop _____
- a metaphor to describe how easy this test is going to be _____
- a metaphor to describe how hard this test is going to be _____

# CS 2200 Spring 2010 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

**Processor design**

2. (9 points, 10 mins)
(a) (3 points)
Given the following load instruction

```
      LW    Rx, Ry, OFFSET    ;     Rx <- MEM[Ry + OFFSET]
```

Show how to realize a new addressing mode, called *indirect*, for use with the load instruction that is represented in assembly language as:

```
      LW    Rx, @(Ry)   ;  Rx <- MEM[MEM[Ry]]
```

The semantics of this instruction is that the contents of register Ry is the address of a pointer to the memory operand that must be loaded into Rx. You should ensure that there are no other side-effects (i.e., only Rx should be affected by this instruction).

(b) (6 points)
Given the following instructions:

```
      ADD   Rx, Ry, Rz  ;     Rx <- Ry + Rz
      ADDI  Rx, Ry, Imm ;     Rx <- Ry + Immediate value
      NAND  Rx, Ry, Rz  ;     Rx <-  NOT (Ry AND Rz)
```

Show how you can use the above instructions to achieve the effect of the following instruction:

```
      CLR   Rx          ;     Rx <- 0
```

You can use an extra register to realize this instruction the contents of which can be trashed.

# CS 2200 Spring 2010 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

3. (10 points, 10 mins)
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
```

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 saving **t** registers on the stack. Fill out the actions similarly for the other entries on the stack (who is responsible for the action caller/callee, and what is the action).

Your answer:

**Stack Pointer→**

| | |
|---|---|
| Local Variables | _____ |
| Saved **s** Registers | _____ |
| Prev Return Address | _____ |
| Add'l Return Values | _____ |
| Add'l parameters | _____ |
| Saved **t** registers | **Caller** saves any t registers whose values it needs upon return |

# CS 2200 Spring 2010 Test 1

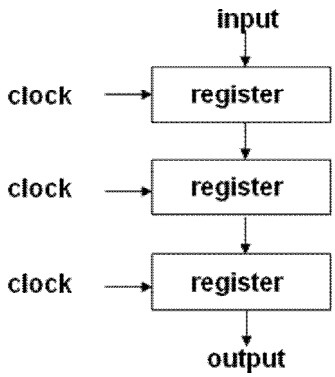
Prism ID:_____

Name:_____ GTID#: 9_____

**Datapath and control**

4. (10 points, 5 mins) (select one)

(a)




How many clock cycles are needed to get the value at the input to the output in the circuit shown:

_____ 0 cycles

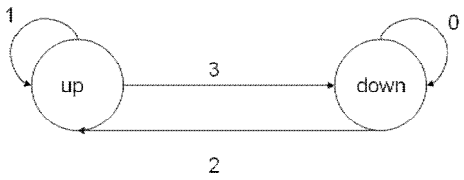
_____ 1 cycles

_____ 2 cycles

_____ 3 cycles

_____ 4 cycles

(b)




The number of rows in the state transition table for the above FSM is:

_____ 2

_____ 3

_____ 4

_____ 5

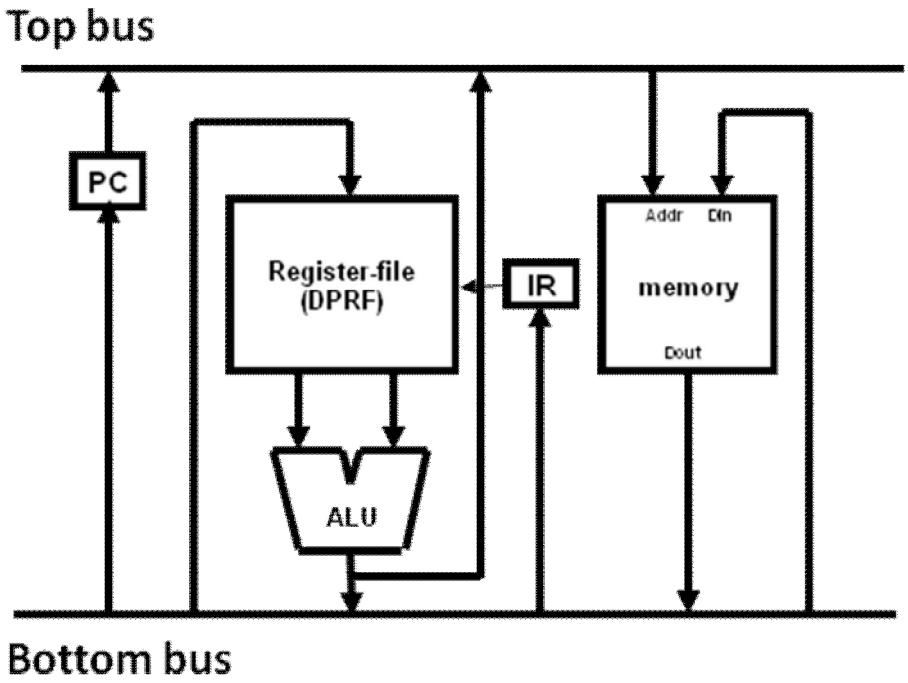
_____ 6

# CS 2200 Spring 2010 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

5. (10 points, 10 min)




Using the above datapath, we wish to perform the following operations:

- IR <- memory[PC]; get contents of memory addressed by PC into IR
- Reg-file[IR 27-24] <- Reg-file[IR 23-20] + Reg-file[IR 3-0]; add values from two registers in the register file and store the result in a third register

Cycle by cycle, show the datapath actions that will accomplish the above operations. Use register transfer format (e.g., ALU -> Bottom bus; Bottom bus -> PC, etc.) to show your work in each cycle.

Prism ID:_____

Name:_____ GTID#: 9_____

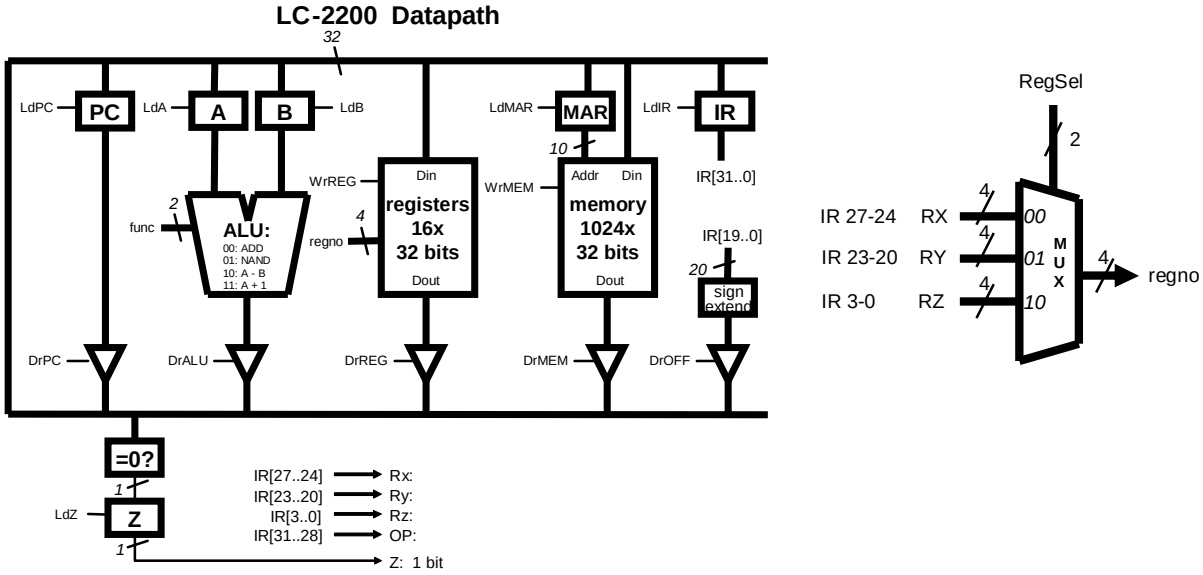
(additional workspace for 5)

# CS 2200 Spring 2010 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

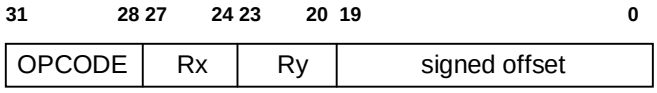
6. (20 points, 15 min)
You are given below a datapath similar to what we have discussed in class.




Given SW instruction as follows:

SW Rx, offset(Ry); MEM[Ry + signed offset] <- Rx;

The instruction format is as shown below:




Write the sequence for implementing the SW (**you don't need to write the fetch sequence for the instruction**). For each microstate, show the datapath action (in register transfer format such as A <- Rx) along with the control signals you need to enable for the datapath action (such as DrREG).

Write your answer on the next page.

Prism ID:_____

Name:_____ GTID#: 9_____

(additional workspace for 6)

# CS 2200 Spring 2010 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

**Interrupts**

7. (15 points, 10 mins)
(a)

**Save *ko* on stack**
**Enable interrupt**
**XXXX**
**Device code**
**Restore state**
**Disable interrupt**
**Restore *ko* from stack**
**Return from interrupt**

In the above interrupt handler code xxxx is

_____ Get interrupt vector from device

_____ Save state

_____ Save vector received from device on stack

_____ Acknowledge interrupt

(b) (10 points, 10 mins)
In the following sentence, use the terms “program counter” , “handler address”, “interrupt vector”, “operating system”, “CPU”, “interrupt vector table”, “data bus”, “program discontinuities”, “interrupt acknowledge”, “low memory” to fill in the blanks.

**Each device has a unique _____ assigned by the _____. At boot time, the operating system builds the _____ in _____ with the handler addresses for all the known _____. Upon receiving an _____ from the _____, the device puts out its interrupt vector on the _____. The CPU uses the interrupt vector as an index into the interrupt vector table to look up the _____ to load into the _____ for handling the interrupt.**

# CS 2200 Spring 2010 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

**Performance**

8. (10 points, 10 min)

Machine **Ma**:

- All LC 2200 instructions take on an average 5 clock cycles
- Floating point addition emulation in software takes 30 LC-2200 instructions (i.e., a total of 150 clock cycles)

Machine **Mb**:

- Floating point addition implemented in hardware takes 10 clock cycles
- Clock cycle time of processor goes up by 10% compared to **Ma**

What fraction of the instructions ought to be floating point additions to guarantee at least a 25% improvement in execution time of **Mb** over **Ma**?

Use the following equations in doing this problem:

$$\text{Execution time} = n * CPI_{Avg} * \text{clock cycle time}$$

$$\text{Improvement in execution time} = \frac{\text{old execution time} - \text{new execution time}}{\text{old execution time}}$$

# CS 2200 Spring 2010 Test 1

Prism ID:_____

Name:_____ GTID#: 9_____

**Pipelining**

9. (15 points, 10 mins)(**SHORT ANSWERS WE MEAN IT!!!!**)
(a) Define *Instruction Level Parallelism*, showing an example code to explain your definition.

(b) What is the most important consideration in designing the stages of a pipelined processor?

(c) Explain the difference between *latency* and *throughput* in the context of a pipelined processor implementation. What are the metrics used for each?

(d) What gives the independence between the stages in the sandwich assembly line that we discussed in class? Identifies the corresponding entities in the instruction pipeline that give the same independence.

(e) Explain the process by which the size and content of the pipeline registers are decided in the instruction pipeline.