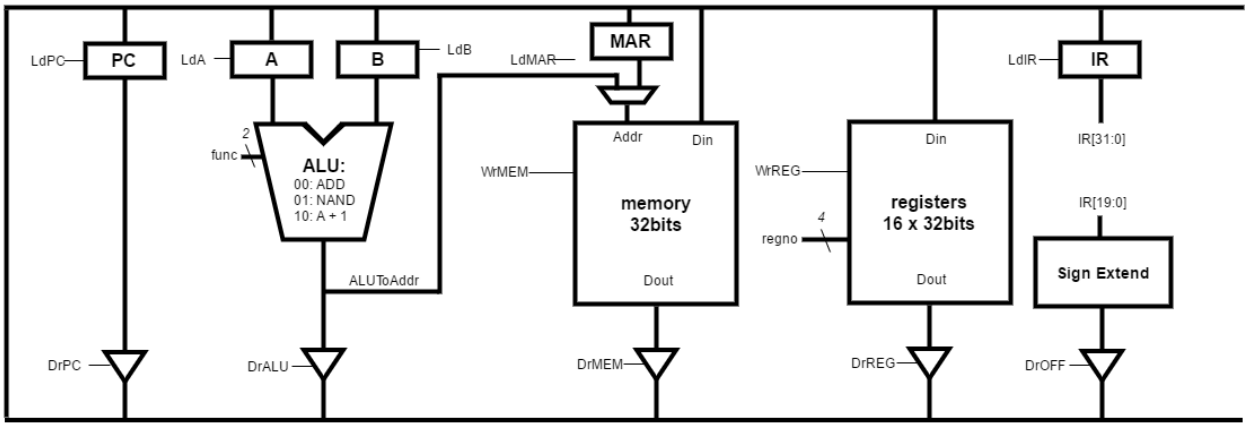
CS 2200 - Homework 2
Spring 2017

Important information:

1. Submit to T-Square as a PDF file.
2. This is an individual assignment. No collaboration is permitted.
3. Due date: February 13, 11:55pm

Problem 1: Datapath

Below is a modified datapath, the LC-2201, including an extra wire that feeds the output from the ALU directly to the address input for memory (ALUToAddr). Points will be taken off for inefficient answers.



1. Give the states to control the **UNMODIFIED** LC-2200 datapath in order to execute the SW instruction. Use the format of the given state below, and only list signals that are asserted.

```
State0:  DrREG, regno = RY, LdA
State1: DrOFF, LdB
State2: func = ADD, DrALU, LdMAR
State3: regno = RX, DrREG, WrMEM
```

2. Give the states to control the **MODIFIED** LC-2201 datapath in order to execute the SW instruction.

```
State0: DrREG, regno = RY, LdA
State1: DrOFF, LdB
State2: func = ADD, regno = RX, DrREG, WrMEM, MemoryMUX = ALUToAddr
```

3. What is the CPI for the SW instruction with and without the DPRF enhancement?

```
Without: 4
With: 3
```

4. Assuming that the clock speed remains the same, what is the speedup for the SW instruction in the LC-2201 datapath above versus the LC-2200 datapath.

```
4/3 = 1.333
```

Problem 2: Performance

1. In the year 2020, both Intel and AMD decide to abandon their work on the x86 and instead develop processors based on the LC-2200 architecture. In order to build a faster processor than their competition each manufacturer makes specific improvements to their implementation.

   - Intel's LC-2200 processor features additional hardware that offers an average speedup of 1.4x on memory accesses (LW and SW operations).
   - AMD's LC-2200 processor features better branch prediction, which gives its processor an average speedup of 1.9x on BEQ operations.

   (a) Using the chart below, calculate the speedup on Intel over AMD of GCC (GNU Compiler Collection) and SPICE (Simulation Program with Integrated Circuit Emphasis) on each manufacturer's processor.

   ```
   Intel GCC 21/100/1.4=0.15, 0.52+0.25+0.15+0.02=0.94;;Intel SPICE 30/100/1.4=0.2142, 0.5+0.17
   +0.03+0.2142=0.9142;;AMD GCC 25/100/1.9=0.21, 0.21+0.02+0.1315+0.52=0.8815;;AMD SPICE
   17/100/1.9 = 0.0894. 0.0894 + 0.5 + 0.3 + 0.03 = 0.9194. A/I GCC = 0.938, A/I SPICE = 1.006
   ```

   (b) Which processor will execute GCC faster?

   ```
   Intel
   ```

   (c) Which processor will execute SPICE faster?

   ```
   AMD
   ```

Instruction Type	LC-2200 Instructions	Percent of Instructions in GCC	Percent of Instructions in SPICE
Arithmetic	ADD, ADDI, NAND	52%	50%
Data Transfer	LW, SW	21%	30%
Branching	BEQ	25%	17%
Jumping	JALR	2%	3%

2. You are trying to decide which of 2 computers to buy, the AMD-2500 **(2.1 GHz)** and Intel-5000 **(1.8 GHz)**. Both architectures only include 3 instructions, and you only intend to run 1 program on these architectures. The number of cycles for each instruction and their frequency in the program are listed below.

	Instruction 1	Instruction 2	Instruction 3
Cycles for AMD	10	5	3
Cycles for Intel	6	3	5
Frequency	30%	50%	20%

   (a) Compute the speedup of AMD over Intel (round to nearest hundredth). Based on this, which processor would you choose to buy?

```
Would buy Intel b/c it is faster.
1a = 10/2.1 0.3 = 1.428
2a = 5/2.1 * 0.5 = 1.19
3a = 3/2.1 * 0.2 * 0.2857
a = 1.428+1.19+0.285 = 2.904
1i = 6/1.8*0.3 = 1
2i = 3/1.8*0.5 = 0.833
3i = 5/1.8*0.2 = 0.556
i = 1+0.833+0.556 = 2.389

Speedup of AMD over Intel = i/a = 2.389/2.904 = 0.83
```

Problem 3: Interrupts

1. What does the **RETI** instruction do? Explain why it is necessary to make **RETI** atomic.

RETI is broken into four parts: 1. Disable interrupts 2. Restore $k0 3. Enable interrupts 4. JALR back to return address. RETI must be atomic because it is composed of multiple operations that must be executed in sequence w/o interrupt(s) within the interrupt.

2. What does the Interrupt Vector Table (IVT) hold? What is the Exception/Trap register used for?

IVtT is populated by the manufacterer and holds all interrupt handlers that are results from their corresponding discontinuities (exceptions, traps and interrupts). Exception/Trap register holds the current interrupt, taken from the IVT.

3. Below is an interrupt handler written in LC-2200 assembly. However, it is missing some crucial lines of assembly code to correctly handle the interrupt. Your task is to fill in the missing pieces. (Refer to your class notes and chapter 4 in the book to learn exactly what sequence of tasks the interrupt handler performs).

Some helpful instructions relevant to interrupts are:

- **EI** (Enable Interrupts)
- **DI** (Disable Interrupts)
- **RETI** (Return from Interrupt)

```
counter_inthandler:
    sw $k0, -3($sp)
    ---------------------------
    ei                      !  Enable interrupts
    sw $s0, -2($sp)
    sw $s1, -1($sp)
    ---------------------------
    la $s0, counter         !  Incrementing counter
    lw $s0, 0($s0)          !  $s0 = 0xFFFFFC
    lw $s1, 0($s0)          !  $s1 = contents of 0xFFFFFC
    addi $s1, $s1, 1
    sw $s1, 0($s0)
    lw $s0, -2($sp)
    ---------------------------
    lw $s1, -1($sp)
    ---------------------------
    DI
    ---------------------------
    lw $k0, -3($sp)
    ---------------------------
    RETI
    ---------------------------
counter:  0xFFFFFC
```