CENG 3420 Homework 2

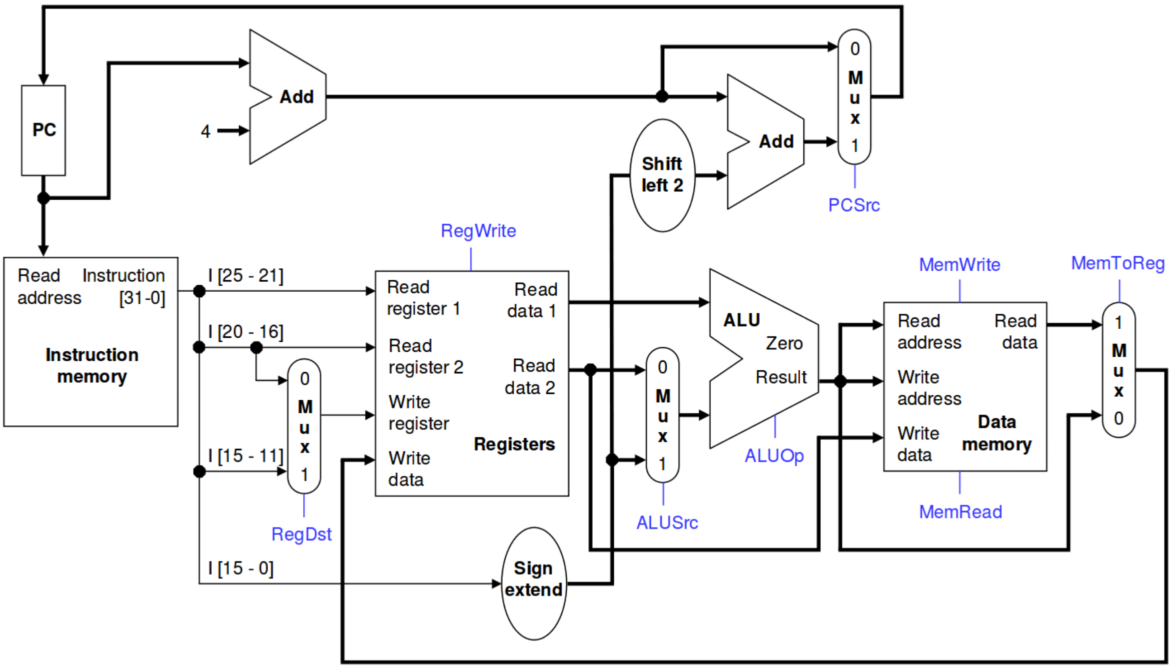
Due: Mar. 14, 2016

Q1 List the drawbacks of single-cycle processor.

Q2 Single-cycle Datapaths:

1. The MIPS instruction `lw $rt, offset($rs)` sets register `$rt` to the value at `Mem[$rs + offset]`. This is an I-type instruction, where offset is a 16-bit immediate. When executed on the single-cycle datapath shown below, the control signals for `lw` are:

inst	PCSrc	ALUSrc	ALUOp	MemWrite	MemRead	MemToReg	RegDst	RegWrite
lw			add					



2. Consider a new R-type instruction `lwd $rd,$rt($rs)` which sets register `$rd` to the value at `Mem[$rs +$rt]`. Show that this instruction can be executed on the above datapath by setting the control signals appropriately:

inst	PCSrc	ALUSrc	ALUOp	MemWrite	MemRead	MemToReg	RegDst	RegWrite
lw								

Q3 List the 5 stages of the pipelined MIPS processor, and show the tasks done in each stage.

Q4 In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

IF	ID	EX	MEM	WB
200ps	100ps	120ps	210ps	150ps

1. What is the clock cycle time in a pipelined and non-pipelined processor?
2. What is the total latency of an `LW` instruction in a pipelined and non-pipelined processor?
3. If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

Q5 The following code is run on a 5-stage MIPS pipeline with full forwarding.

```
lw   $t0, 0($a0)
addi $t0, $t0, 1
sw   $t0, 0($a0)
addi $a0, $a0, 4
```

1. Identify all the data hazards that cannot be resolved with forwarding.
2. Rewrite the code to eliminate stalls on the 5-stage pipeline with full forwarding.

Q6 In this exercise we look at memory locality properties of matrix computation. The following code is written in C, where elements within the same row are stored contiguously.

```
for (J=0; J<8000; J++)
  for (I=0; I<8; I++)
    A[I][J]=B[J][0]+A[J][I];
```

1. How many 32-bit integers can be stored in a 16-byte cache line?
2. References to which variables exhibit temporal locality?
3. References to which variables exhibit spatial locality?

Q7 For a direct-mapped cache design with a 32-bit address, the following bits of the address are used to access the cache.

Tag	Index	Offset
31-12	11-6	5-0

1. What is the cache line size (in words)?
2. How many entries does the cache have?

3. What is the ratio between total bits required for such a cache implementation over the data storage bits?

Q8 Media applications that play audio or video files are part of a class of workloads called streaming workloads; i.e., they bring in large amounts of data but do not reuse much of it. Consider a video streaming workload that accesses a 512 KB working set sequentially with the following address stream:

2, 4, 6, 8, 10, 12, 14, 16, ...

1. Assume a 64 KB direct-mapped cache with a 32-byte line. What is the miss rate for the address stream above? How is this miss rate sensitive to the size of the cache or the working set? How would you categorize the misses this workload is experiencing, based on the 3C model?
2. Re-compute the miss rate when the cache line size is 16 bytes, 64 bytes, and 128 bytes. What kind of locality is this workload exploiting?
3. "Prefetching" is a technique that leverages predictable address patterns to speculatively bring in additional cache lines when a particular cache line is accessed. One example of prefetching is a stream buffer that prefetches sequentially adjacent cache lines into a separate buffer when a particular cache line is brought in. If the data is found in the prefetch buffer, it is considered as a hit and moved into the cache and the next cache line is prefetched. Assume a two-entry stream buffer and assume that the cache latency is such that a cache line can be loaded before the computation on the previous cache line is completed. What is the miss rate for the address stream above?