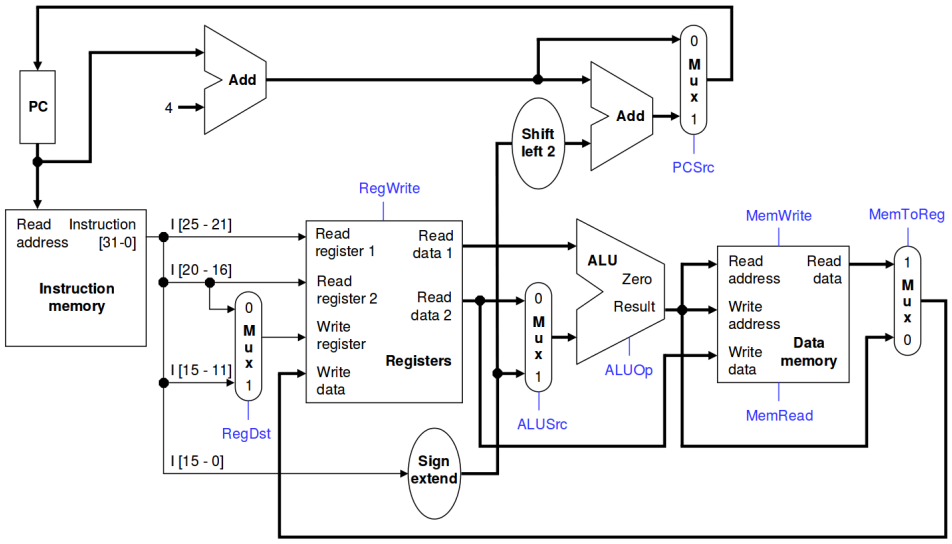
CENG 3420 Final (2018 Spring)

Name: _____
ID: _____

Q1 (**15 %**) Short answer questions. **Circle** the right answer.

1. **Latch / Flip-Flop** is edge triggered, thus can be inserted between pipeline stages.
2. The byte address of MIPS is **Big / Little** Endian.
3. Stack pointer `$sp` is used to address the stack. When a data is added onto the stack, `$sp` ← **`$sp+4` / `$sp-4`**.
4. A conventional adder structure is Ripple Carry Adder (RCA), which is **fast / slow**. RCA consumes **lots / little** energy due to the impact of *glitching*.
5. We can detect an overflow by: Carry into MSB **xor / or / and** Carry out of MSB.
6. Page Mode DRAM operation can reduce redundant **row / column** address parsing.
7. **DRAM / SRAM** cell is written by charge or discharge capacitor.
8. In a memory system, increasing `Tag` field length can **increase / decrease** associativity.
9. Virtual memory always use **write-back / write-through**, as it writes with speed of cache.
10. To detect an I/O device, there are two methods: Port-mapped I/O (PMIO) and Memory-mapped I/O (MMIO). MIPS uses **PMIO / MMIO**.
11. There are several types of data hazards: Read After Write (RAW), Write After Read (WAR), Write After Write (WAW). **RAW / WAR / WAR** is true data dependency.
12. VLIW is **Static / Dynamic** multiple-issue processor.
13. Shared Memory Multiprocessor (SMP) usually uses **spin-locks / message passing** for synchronization.
14. **NUMA / UMA** can scale to larger sizes and have lower latency to local memory.

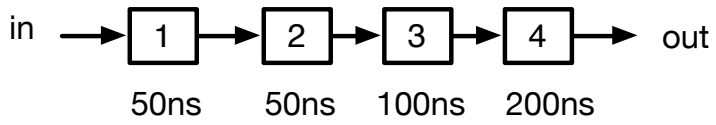
Q2 (5%)



Based on the above datapath, finish the blanks in the following table. Here `lwd` is a new instruction. `lwd $rd,$rt($rs)` sets register `$rd` to the value at `Mem[$rs +$rt]`.

	RegDst	ALUSrc	MemReg	RegWr	MemRd	MemWr
`lw`	0		1		1	0
`sw`	X		X		0	1
`beq`	X		X		0	0
`lwd`		0		1		

Q3 (**10%**) Given a pipeline with each stage latency labelled in below figure, please answer the following questions.



1. What is throughput when input 10 instructions?
2. Which stages are bottleneck?
3. Please give a pipeline to improve throughput.
4. What is throughput for your designed pipeline?

Q4 (**12%**) For a direct-mapped cache design with a 32-bit address and byte-addressable memory, consider the following two address configurations to access the cache:

	Tag	Index	Block offset	Byte offset
configuration a	$31-10$	$9-5$	$4-2$	$1-0$
configuration b	$31-12$	$11-6$	$5-2$	$1-0$

1. For both configurations, what's the cache block size (in words)? How many blocks does the cache have?
2. How many total bits are required to implement each cache.

Q5 (**8%**) Assume it takes one clock to send address to DRAM memory and one clock to send data back. DRAM has 7 cycle latency for first byte, and 3 cycles for each of subsequent bytes in the block. To transfer a 8-byte block, calculate the cycle number if we need:

1. non-interleaving;
2. 2-module interleaving.

Q6 (**10%**) A processor with frequency 40 MHz performs a standard test program, whose instruction type, instruction number and CPI are shown in the following Table:

Instruction Type	# Instruction	CPI
Integer Operation	45000	1
Data Transmission	35000	2
Float Operation	15000	2
Transmission Control	5000	2

1. What is the average Cycles Per Instruction (CPI) of this program?
2. What is the Instruction Per Second (IPS) of this program?
3. What is total run time for performing this program?

Q7 (**10%**) Consider the following portions of two different programs running at the same time on four processors in a share memory multiprocessor (SMP). Assume that before this code is run, both x and y are 0.

Core1: `x = 3;`

Core2: `y = 3;`

Core3: `w = x + y + 1;`

Core4: `z = x - y;`

Core5: `r = w + z;`

1. What are all the possible resulting values of w, x, y, z, and r? For each possible outcome, explain how we might arrive at those values.
2. How could you make the execution more deterministic so that only one set of values is possible?

Q8 (**15%**) Problems in this exercise refer to the following instruction sequences:

```
lp: lw   $t0, 0($s1)
    addu $t0, $t0, $s2
    sw   $t0, 0($s1)
    addi $s1, $s1, -4
    bne  $s1, $0,  lp
```

1. Find all data dependences in this instruction sequence.
2. Find all hazards in this instruction sequence for a 5-stage pipeline with and then without forwarding.
3. Eliminate unnecessary loop overhead instructions, in compiling stage we can use **Loop Unrolling** technique. Please give new instruction sequence using 4 times loop unrolling.

Q9 (**15%**) We will examine space/time optimizations for page tables. The following Table shows parameters of two virtual memory systems. For each system, please answer the following questions. (note that 1 byte = 8 bits, KB is KiloBytes and Kb is Kilobits)

	Virtual Address (bits)	Physical DRAM Installed	Page Size	PTE Size (byte)
a	32	4 GB	8 KB	4
b	64	16 GB	4 KB	8

1. For a single-level page table, how many page table entries (PTEs) are needed?
2. How much physical memory is needed for storing the page table?
3. Using a multilevel page table can reduce the physical memory consumption of page tables, by only keeping active PTEs in physical memory. How many levels of page tables will be needed?
4. How many memory references are needed for address translation if missing in TLB?