CENG 3420 Midterm (2019 Spring)

Name: _____
ID: _____

Solutions

Q1 (10%) Short answer questions. **Circle** the right answer.

1. Intel x86 is **RISC / CISC**.
2. In MIPS instruction set architecture, instruction `add` and `sub` are **R / J** format.
3. A conventional adder structure is Ripple Carry Adder (RCA), which is **fast / slow**.
4. 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. The time between the start and the completion of a task is **Response time / Throughput**.
7. Stack pointer `$sp` is used to address the stack. When a data is added onto the stack, `$sp` ← **`$sp+4 / $sp-4`**.
8. **Latch / Flip-Flop** is edge triggered, thus can be inserted between pipeline stages.
9. In pipeline, forwarding is a technique to resolve **structure / data** hazards.
10. Pipeline clock rate is limited by the **Fastest / Slowest** stage.

A1
1. CISC
2. R
3. slow
4. lots
5. xor
6. Response time
7. `$sp-4`
8. Flip-Flop
9. data
10. slowest

Q2 (10%) Assume `$t0=0xAAAAAAAA, $t1=0x12345678`. Find the value of `$t2` after the following instructions, respectively.

1.
```
sll $t2, $t0, 4
and  $t2, $t2, $t1
```

2.
```
sll  $t2, $t0, 4
addi $t2, $t2, 1
```

3.
```
srl  $t2, $t0, 3
andi $t2, $t2, 0xFFEF
```

A2 As shown below,

1. 0x02200220
2. 0xAAAAAAA1
3. 0x00005545

Q3 (10%) You are required to develop some simple measures of pipeline performance and relative speedup.

1. Let $T_{k,n}$ be the total time required for a pipeline with k stages to execute n instructions. Speedup of k stage pipeline is given by,

$$S_k = \frac{T_{1,n}}{T_{k,n}}. \tag{1}$$

   Determine S_k in terms of k and n.

2. Consider an instruction sequence of length n that is streaming through the instruction pipeline. Let p be the probability of encountering a conditional or unconditional branch instruction, and let q be the probability that execution of a branch instruction I causes a jump to a nonconsecutive address. Assume that each such jump requires the pipeline to be cleared, destroying all ongoing instruction processing, when I emerges from the last stage. Determine S_k in terms of k, n, p and q.

A3 1.

$$S_k = \frac{nk}{k+n-1}.$$

2.

$$S_k = \frac{nk}{pqnk + (1-pq)(k+n-1)}.$$

Q4 (10%) A program runs in $10s$ on computer A with 2GHz clock. If we want to design a computer B such that the same program can be finished in $7s$, determine the clock frequency of computer B. Assume it requires only 0.7X clock cycles to execute the program on computer B due to different CPU design.

A4 CPU clock cycle of the program on computer A is,

$$cycle_A = 10s \times 2GHz = 2 \times 10^{10}\text{cycles}. \tag{2}$$

CPU clock cycle of the program on computer B is,

$$cycle_B = 0.7 \times cycle_A = 1.4 \times 10^{10}\text{cycles}. \tag{3}$$

Clock frequency of computer B will be,

$$cycle_B/7s = 2GHz. \tag{4}$$

Q5 (10%) An assembly function dealing with an array `array[]` with size `size` can be found as follows. Assume two parameters `array` and `size` are stored in `$a0` and `$a1`. Please explain the functionality of the assembly code.

```
move $t0, $zero
loop:
sll $t1, $t0, 2
add $t2, $a0, $t1
sw $zero, 0($t2)
addi $t0, $t0, 1
slt $t3, $t0, $a1
bne $t3, $zero, loop
```

A5 The assembly code clears the array/ Set every elements in the array to 0.

Q6 (15%) Given the following specs of the datapath latencies:

Stages	IF	ID	EX	MEM	WB
Latencies (ps)	250	350	150	300	200

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?

A6
1. Non-pipelined: 1250*ps*; Pipelined: 350*ps*.
2. Non-pipelined: 1250*ps*; Pipelined: 1750*ps*.
3. Split ID stage. New clock cycle will be 300*ps*.

Q7 (10%) Regarding the following MIPS instruction:

```
sw R10, -20(R6)
```

1. Which registers need to be read?
2. What does this instruction do in the EX and MEM stages?

A7
1. `R6` and `R10` need to be read.
2. EX stage: -20 + `R6`; MEM stage: write value to memory.

Q8 (10%) Regarding the following instructions:

```
I1:  or  R1, R2, R3
I2:  or  R2, R1, R4
I3:  or  R1, R1, R2
```

1. Indicate dependencies and their type.
2. Assume there is full forwarding. Indicate hazards and add `nop` instructions to eliminate them.

A8
1. RAW: `R1` from `I1` to `I2`, `I3`; `R2` from `I2` to `I3`. WAR: `R2` from `I1` to `I2`; `R1` from `I2` to `I3`. WAW: `R1` from `I1` to `I3`.
2. No RAW hazard because of the existence of forwarding.

Q9 (15%) Given the following loop, assume that perfect branch prediction is used (no stalls due to control hazards), that there are no delay slots, and that the pipeline has full forwarding support. Also assume that many iterations of this loop are executed before the loop exits.

```
loop:  lw R1, 0(R1)
       and  R1, R1, R2
       lw   R1, 0(R1)
       lw   R1, 0(R1)
       beq R1, R0, loop
```

1. Show a pipeline execution diagram for the third iteration of this loop, from the cycle in which we fetch the first instruction of that iteration up to (but not including) the cycle in which we can fetch the first instruction of the next iteration. Show all instructions that are in the pipeline during these cycles (not just those from the third iteration).
2. How often (as a percentage of all cycles) do we have a cycle in which all five pipeline stages are doing useful work?
3. At the start of the cycle in which we fetch the first instruction of the third iteration of this loop, what is stored in the IF/ID register?

A9 1. See the Figure 1:

LW R1,0(R1)	WB
LW R1,0(R1)	EX MEM WB
BEQ R1,R0,Loop	ID *** EX MEM WB
LW R1,0(R1)	IF *** ID EX MEM WB
AND R1,R1,R2	IF ID *** EX MEM WB
LW R1,0(R1)	IF *** ID EX MEM
LW R1,0(R1)	IF ID ***
BEQ R1,R0,Loop	IF ***

Figure 1: Answer of Q8-1

2. 0 %.
3. The address of that first instruction of the third iteration ($PC+4$ for the BEQ from the previous iteration) and the instruction word of the BEQ from the previous iteration.