**University of California at Merced**

**CSE 140 Computer Architecture**

**Midterm (Summer 2021)**

**Your NAME: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Instructions:**

1. This is a closed book and closed notes exam (except for an access to the MIPS\_Reference\_Sheet).
2. Besides a laptop/desktop to access/complete the midterm, no other computing/electronic devices are allowed. Please turn off/silence cell phones.
3. Answer as many questions as possible. Partial credit will be given where appropriate.
4. Answer all questions on your own.
5. Be sure to clearly indicate your final answer for each question. Also, be sure to state any assumptions that you are making in your answers.

Good luck!

|  |  |  |
| --- | --- | --- |
| **Problem** | **Possible Score** | **Your Score** |
| **1** | **10** |  |
| **2** | **15** |  |
| **3** | **13** |  |
| **4** | **16** |  |
| **5** | **28** |  |
| **6** | **18** |  |
| **Total** | **100** |  |

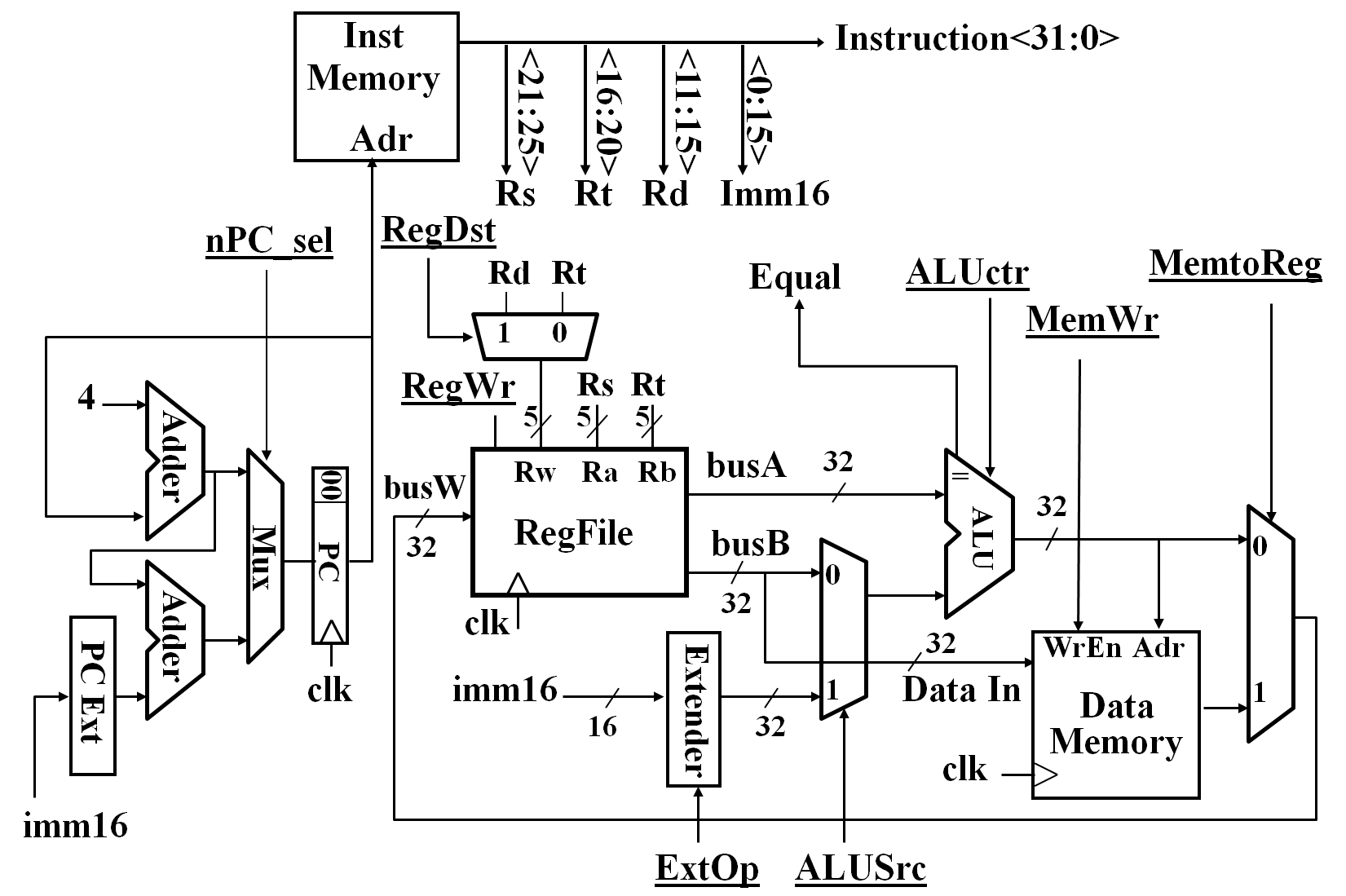
1. [10 pts] Indicate whether the following statements are true or false:

|  |  |  |
| --- | --- | --- |
| 1. | In ***beq***, ***label*** can be somewhere in an offset of from PC+4. | T / F |
| 2. | In a **J** instruction, the target address is calculated by adding 2 zeros to the left and the first 4 bits of PC to the address field. | T / F |
| 3. | We can always reduce miss rate of a cache by increasing the number of sets, due to temporal locality. | T / F |
| 4. | A pipelined processor allows multiple instructions to share the same datapath. | T / F |
| 5. | Miss penalty can be reduced by increasing the cash size. However, the cost of the cash will increase. | T / F |

2. [15 pts] Translate the following instruction from machine code to a MIPS instruction with proper register names.

0x0211402A

3. [13 pts] **Trace** through the paths the execution of **the instruction from problem 2** in this single cycle datapath design. Fill in the appropriate control signal values in the table.



(Zeros(0) or Signed(1))

1

0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| nPc\_sel | ExtOp | ALUsrc | ALUctr | MemWr | MemtoReg | RegDst | RegWr |
|  |  |  |  |  |  |  |  |

4. Given the 5 stages of a processor with the following latencies:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IF | ID | EXE | MEM | WB |
| 150ps | 50ps | 350ps | 300ps | 150ps |

Assume that when pipelining, each pipeline stage costs extra **10ps** for the registers between pipeline stages.

1. [2 pt] Without pipelining: What is the cycle time of the processor?
2. [2 pt] Without pipelining: What is the latency of an instruction for the processor?
3. [2 pt] With pipelining: What is the cycle frequency of the processor?
4. [2 pt] With pipelining: What is the latency of an instruction for the processor?
5. [2 pt] 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?
6. [6 pt] Do you think it is a good idea to split the stages as described in **part e**? Why? Justify your answer in terms of cycle time, latency, and throughput (with pipelining).

5. Given the MIPS instructions below:

1 add s1, s0, t0

2 lw s0, 4(t0)

3 add t2, s0, s1

4 slt s1, s0, t2

5 sub t4, s1, t2

6 add s0, t4, s1

1. [8 pts] List all the RAW (read-after-write) data dependencies (NOT hazards). You must indicate the line numbers and register name in this format: **#x --> #y / $z** (line x depends on line y with register z).
2. Assume **only ALU-ALU forwarding support**, no delay slot, and register files **can** perform read and write at the same cycle.
   1. [6 pts] Show a pipeline execution diagram. Make sure to align each stage neatly. (use “x” to show stalls). Some columns might remain empty depending on the final number of cycles.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CYCLE number= | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| ADD | I | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LW |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ADD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SLT |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SUB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ADD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* 1. [4 pts] What is the total execution time (in cycles) of running this code?

c. [10 pts] Repeat part b **without any forwarding support**, no delay slot, and register files **CAN** perform read and write at the same cycle. (answer both i. and ii)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CYCLE number= | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| ADD | I | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LW |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ADD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SLT |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SUB |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ADD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

6. [18 pt] Given a 2-way set associative cache of 64 Bytes, with block size = 8 bytes, LRU replacement policy, fill in the table of cache access in the following cache settings:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Tag (in binary) | Index (in binary) | Offset (in binary) | Hit/Miss |
| 8 |  |  |  |  |
| 49 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |
| 28 |  |  |  |  |
| 93 |  |  |  |  |
| 125 |  |  |  |  |
| 31 |  |  |  |  |