David Wang
CSE 140
Collab w/ Jan Tanja

CSE 140 HW #3

Solve the following problems and place your answers in a text document for submission.

1. Given the following MIPS code snippet (note that instruction #6 could be anything):

```
loop:
1     addi $t0, $t0, 4
2     lw $v0, 0($t0)
3     sw $v0, 20($t0)
4     lw $s0, 60($t0)
6     bne $s0, $0, loop
7     ##  The following instruction could be anything!
```

a) Detect hazards and insert no-ops to insure correct operation. Assume no delayed branch, no forwarding units and no interlocked pipeline stages. Your answer on the right should take the form of pair(s) of numbers: num@location – indicating num no-ops should be placed at location. E.g., if you wanted to place 6 noops between lines 2 and 3 (i.e., location=2.5) and 8 noops between lines 5 and 6 (i.e., location=5.5), you would write: "6@2.5, 8@5.5".

2@1.5

2@2.5

2@4.5

b) Now, reorder/rewrite the program to maximize performance. Assume delayed branch and forwarding units, but no interlocked pipeline stages. For unknown reasons, the first instruction after the loop label must be the addi. Feel free to insert no-ops where needed. You should be able to do it using 6 instructions per loop (10 pts) or only 5 instructions (hard, 15 pts).

_____ ##Extra instruction before loop (if any)

_____ ##Extra instruction before loop (if any)

loop:

```
1     addi $t0, $t0, 4
2     lw $v0, 0($t0)
3     lw $s0, 60($t0)
4     sw $v0, 20($t0)
5     bne $s0, 0, loop
6     no-op
7     ##  The following instruction could be anything!
```

c) Name three reasons why we use caches in a pipelined processor

Caches are useful in a pipelined processor because they can do expensive computations repeatedly, are faster than memory, and have associativity with block size and size of cache

d) Circle whether each change will make something decrease (-), increase (+) or remain the same (=). The effects you will consider are the number of instructions in the program, the amount of cycles for each instruction (in a pipelined processor) and the amount of time for each cycle (seconds/cycle).

	Instructions/Program	**Cycles/Instruction**	**Seconds/Cycle**
Reducing the number of registers in the ISA	- = **+**	- = **+**	**-** = +
Adding a branch delay slot	- = **+**	**-** = +	- **=** +
Merging the Execute and Mem stages (loads and stores use an additional adder to calculate base+offset)	**-** = +	**-** = +	- **=** +
Changing the implementation from a microcoded CISC machine to a RISC pipeline	- **=** +	**-** = +	- **=** +

2. Consider the following loop.

```
loop:     lw   r1, 0(r1)
          and  r1, r1, r2
          lw   r1, 0(r1)
          lw   r1, 0(r1)
          beq  r1, r0, 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.

a) 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).

LW r1, 0,(r1) WB:

BEQ r1, r0, LOOP: ID EX MEM WB

3: LOOP: LW r1, 0(r1): IF IF EX MEM WB

3 AND r1, r1, r2 IF * ID EX MEM WB
3 LW r1, 0(r1) IF ID EX MEM WB
3 LW r1, 0(r1) IF * ID EX MEM WB
3 BEQ r1, r0, LOOP IF ** ID EX MEM WB

b) How often (as a percentage of all cycles) do we have a cycle in which all five pipeline stages are doing useful work?

3/5 Instructions = 60% of all cycles

3. Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 32-bit memory address references, given as word addresses (addresses are assigned according to words, not bytes): **3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253**
    a. For each of these references, identify the binary address (in words, not bytes), the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty and the references are accessed according to the order as listed. The first address has been done as an example.

Word Address	***Binary Address***	***Tag***	***Index***	***Hit (H) / Miss (M)***
3	0000 0011	0	3	M
180	1011 0100	11	4	M
43	0010 1011	2	11	M
2	0000 0010	0	2	M
191	1011 1111	11	15	M
88	0101 1000	5	8	M
190	1011 1110	11	14	M
14	0000 1110	0	14	M
181	1010 1101	10	13	M
44	0010 1100	2	12	M
186	10111010	11	10	M
253	1111 1101	15	13	M

    b. For each of these references, identify the binary address (in words), the tag, and the index given a direct-mapped cache with two-word blocks and a total size of 8 blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

Word Address	***Binary Address***	***Tag***	***Index***	***Hit (H) / Miss (M)***
3	0000 0011	0	1	M
180	0000 0011	0	0	M
43	0000 0011	0	1	M
2	0000 0011	0	1	H
191	0000 0011	0	7	M
88	0000 0011	0	4	M
190	0000 0011	0	7	M
14	0000 0011	0	7	M
181	0000 0011	0	6	M
44	0000 0011	0	6	M
186	0000 0011	0	5	M
253	0000 0011	0	6	M

c. You are asked to optimize a cache design for the above references. There are three direct-mapped cache designs possible, all with a total of 8 words of data: **C1** has 1-word blocks, **C2** has 2-word blocks, and **C3** has 4-word blocks. If the miss stall time is 25 cycles, and C1 has an access time of 2 cycles, C2 takes 3 cycles, and C3 takes 5 cycles, which is the best cache design? (Use a table, similar to part a and b, to help you evaluate the hit/miss rate of C1, C2, and C3.)

Word Addresses	Binary Addresses	Tag	C1	H/M	C2	H/M	C3	H/M
3	0000 0011	0	3	M	1	M	0	M
180	1011 0100	11	4	M	0	M	1	M
43	0010 1011	2	11	M	1	M	2	M
2	0000 0010	0	2	M	1	H	0	H
191	1011 1111	11	15	M	7	M	3	M
88	0101 1000	5	8	M	4	M	2	M
190	1011 1110	11	14	M	7	H	3	H
14	0000 1110	0	14	M	7	M	3	M
181	1010 1101	10	13	M	6	M	3	M
44	0010 1100	2	12	M	6	M	3	M
186	1011 1010	11	10	M	5	M	2	M
253	1111 1101	15	13	M	6	M	3	M

C2 is the fastest

d. Below are listed parameters for different direct-mapped cache designs:

Cache Data Size: 32 KiB
Cache Block Size: 2 words
Cache Access Time: 1 cycle

Generate a series of read requests that have a lower miss rate on a 2 KiB 2-way set associative cache than the cache listed above. Identify one possible solution that would make the cache listed have an equal or lower miss rate than the 2 KiB cache. Discuss the advantages and disadvantages of such a solution.

2^(index bits) * (valid bits + tag bits + (data bits * 2 ^ offset bits))

total bits = 2^15 (1 + 14 + (32 * 2^1)) = 2588672 bits

total bits = 2^13 (1 + 13 + (32 * 2^4)) = 4308992 bits

The next smallest cache with 16 word blocks and a 32 bit address is 215892 bits, smaller than the first cache.

4. 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*
31-10	9-5	4-0

a. What is the cache block size (in words)?

2^offset bits = 2^5 bytes = 2^3 words = 8 words

b. How many entries does the cache have?

2^index bits = 2^5 lines

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

(128 * (32 * 8 + 20 + 1)/(128 * (32 * 8)) = 277/256

Starting from power on, the following **byte-addressed** cache references are recorded:
0, 4, 16, 132, 232, 160, 1024, 30, 140, 3100, 180, 2180

d. How many blocks are replaced?

Address	Block Address	Cache Index	H/M
0	0	0	M
4	0	0	H
16	0	0	H
128	4	4	M
224	7	7	M
160	5	5	M
4100	128	0	M
30	0	0	M
140	4	4	H
3100	96	96	M

e. What is the hit ratio?

3/10