



The University of Michigan - Department of EECS

EECS370 – Introduction to Computer Architecture

Midterm Exam 2

April 1, 2008

Name: Key (answers in red)_____

University of Michigan uniqname (Not your ID): _____

Open book, open notes. No laptops, PDAs, cell phones, etc. (calculators are ok). This exam has 11 questions, 9 pages, and 100 points. Questions vary in difficulty; it is strongly recommended that you do not spend too much time on any one question. For questions where a box is provided, please put your final answer in the box.

The rules of the Honor Code of the University of Michigan - College of Engineering apply for this exam. Honor code pledge: I have neither given nor received aid on this examination, nor have I concealed any violations of the Honor Code.

Signature: _____
(Exams without a signed pledge will not be graded)

Question	Score
1	/3
2	/3
3	/4
4	/3
5	/8
6	/6
7	/10
8	/12
9	/21
10	/15
11	/15
Total	/100

Part I - Multiple Choice and Short Answer [27 pts]

1. Assume we have a 2-way superscalar pipeline created by doubling the datapath of the standard LC2k 5-stage pipeline. If branches are resolved in the MEM stage, how many instructions must be squashed in the **worst-case** when a branch is mispredicted? [3 pts]

- a. 3 instructions
- b. 4 instructions
- c. 5 instructions
- d. 6 instructions
- e. 7 instructions**

2. Which of the following statements is true? [3 pts]

- a. For a given cache size, an 8-way associative cache will have a faster cache access time than a 4-way associative cache.
- b. There is no way to reduce compulsory misses.
- c. With separate caches for data and instructions, the data cache will always have better spatial locality than the instruction cache.
- d. Writeback caches always outperform write through caches
- e. a and c are true
- f. All of the statements are true.
- g. None of the statements are true**

3. Assume we have the 5 stage LC2K pipeline used Project 3. In the following program segment, how many read-after-write (RAW) dependences and hazards are present? [4 pts]

add	1	2	3
nand	4	5	6
lw	3	6	4
sw	6	6	6
lw	2	6	3
add	1	3	6

Number of hazards: _____**3**_____

Number of dependences: _____**4 (or 6)**_____

There are 4 dependences, 3 of which are hazards. An answer of 6 dependences, which double counted the dependences for register 6, was also accepted.

- 1. Register 3 from instruction 1 to 3, also a hazard
- 2/3. Register 6 from instruction 3 to 4 for both regA and regB, also both hazards
- 4. Register 3 from instruction 1 to 6
- (5/6. Register 6 from instruction 2 to 4 for both regA and regB)

4. Compare the LC2K single-cycle (SC) datapath, multi-cycle (MC) datapath and the pipelined datapath. Explain why in the average case MC performs better than SC but worse than the pipelined datapath. Answer in 20 words or less. [3 pts]

MC vs SC: shorter clock period (accepted: higher clock frequency, faster clock, etc.)

MC vs Pipeline: less parallelism (accepted: higher CPI, less throughput, not simultaneous, etc.)

5. Consider the following block of code running on a 5-stage LC2K pipeline. Assume that the register file supports internal forwarding, so if a register is read and written on the same cycle, the new value is read out. However, this pipeline has no hazard detection and no data forwarding paths. Reorder the following LC2K instructions to **maximize** performance, adding noops as necessary to ensure correct execution. [8 pts]

```
add 1 2 3
add 2 4 6
lw 0 1 loc1
sw 0 1 loc2
nand 3 2 2
nand 4 2 1
```

```
add 1 2 3
add 2 4 6
lw 0 1 loc1
nand 3 3 2
noop
sw 0 1 loc2
nand 4 2 1
```

```
add 1 2 3
lw 0 1 loc1
add 2 4 6
nand 3 2 2
sw 0 1 loc2
noop
nand 4 2 1
```

6. Consider the following block of code running on a modified MIPS pipeline. In an attempt to improve the clock frequency of the design, an extra execute stage is added, EX2, giving the pipeline a total of 6 stages. Results of the ALU are only available after EX2. Assume this pipeline has full data forwarding paths and stalls when a decoded instruction needs a register value that isn't yet available. [6 pts]

- 1) lw \$1, 100(\$0)
- 2) add \$3, \$1, \$2
- 3) add \$4, \$2, \$3
- 4) sw \$4, 100(\$0)
- 5) add \$1, \$2, \$4
- 6) add \$3, \$5, \$1

What instructions are in which stage at the end of the 7th cycle? Assume the first instruction finishes the IF stage at the end of cycle 1.

IF	ID	EX	EX2	MEM	WB
<u> 4 </u>	<u> 3 </u>	<u> nop </u>	<u> 2 </u>	<u> nop </u>	<u> nop </u>

Part II – Longer Problems [73 pts]

7. Branch prediction [10 pts]

You are considering the following branch predictors for the function `foo()` below. Compute the number of misses for each predictor type on each invocation of `foo()`. Assume that no predictions are necessary for function calls to either `foo()` or `printf()`, and the 2 branches do not interfere with one another in the branch predictor.

- i. predict not-taken
- ii. predict taken
- iii. backward taken, forward not-taken
- iv. 1-bit counter (predict direction taken last time, initially predicts not-taken)

```
void foo() {
    int i = 0;
loop:
    if (i%2 != 0) goto skip;
    printf("Even\n");
skip:
    i++;
    if (i<10) goto loop;
}
}
```

Predictor type	Number of <u>Mis</u> predictions
predict not-taken	$5 + 9 = 14$
predict taken	$5 + 1 = 6$
backward taken, forward not-taken	$5 + 1 = 6$
1-bit counter	$9 + 2 = 11$

Here is the branching activity for each value of `i` (note that `i` changes between branches):

<code>i</code>	0	1	2	3	4	5	6	7	8	9	10
<code>i%2!=0</code>	N	T	N	T	N	T	N	T	N	T	
<code>i<10</code>		T	T	T	T	T	T	T	T	T	N

8. Performance [12 points]

After many grueling years, you finally graduate from UM and find a job with T.A.Electronics. Your first task is to determine the performance of a new line of microprocessors – the TA1000. The TA1000 is a 5-stage pipelined architecture identical to the LC2K architecture you learned in your EECS 370 lecture. It has full data forwarding capabilities except for LW which incurs a 1 cycle delay whenever there are LWs followed by an immediate use. Branches follow an always-not-taken policy and have a 3 cycle penalty when mispredicted.

The *grinder* application run on the TA1000 has the following instruction frequencies:

45% R-type 20% Branches 15% Loads 20% Stores

In *grinder*, 40% of the branches are taken and 50% of all LWs are followed by an immediate use.

A. Assuming all memory accesses require a single cycle, what is the CPI of the TA1000 running *grinder*? [4 pts]

$$\text{CPI} = 1 + 0.4 * 0.2 * 3 + 0.5 * 0.15 * 1 = 1.315$$

B. Now your manager wants you to compare 2 more realistic designs:

Option 1: An I-cache and a D-cache with single cycle access latency are added into the baseline design in part (A). The I-cache has a miss rate of 3% and the D-cache has a miss rate of 6% (no overlapping of misses). On a miss, the main memory is accessed and has a latency of 24 cycles. The clock period is 2ns.

Option 2: The pipeline is expanded to 8 stages increasing the LW-use distance to 2 cycles and resolving branches in stage 7. The clock period is 1.2ns. The cache sizes (and thus the miss rates) remain the same, but the main memory latency grows to 30 cycles.

Which option has higher performance on *grinder*? Justify your answer quantitatively. [8 pts]

$$\text{CPI}_1 = 1.315 + 0.03 * 1.00 * 24 + 0.06 * 0.35 * 24 = 2.539$$

$$\text{TPI}_1 = 2.539 * 2\text{ns} = 5.08$$

$$\text{CPI}_2 = 1 + 0.4 * 0.2 * 6 + 0.5 * 0.15 * 2 + 0.03 * 1.00 * 30 + 0.06 * 0.35 * 30 = 3.16$$

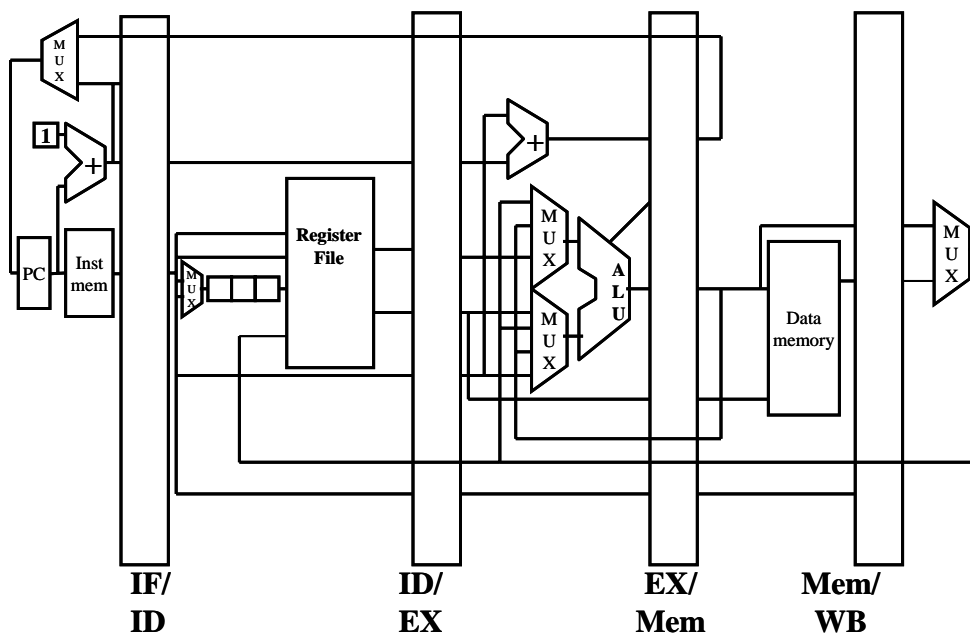
$$\text{TPI}_2 = 3.16 * 1.2\text{ns} = 3.79$$

Hence, 8-stage pipeline is faster than the 5-stage pipeline by ~25%

9. Pipeline datapath design [21 pts]

You are hired as a consultant by a new upstart semiconductor company AMC to help create a high-speed pipelined LC2k implementation – the Pacer. AMC has figured out how to push clock speeds of their processor to new levels. However, the instruction and data caches can only run at half speed. You discover that while the caches are slow, they can read/write **2 consecutive words** at a time. Note that each cache can only process a single request at a time. The objective is to create an LC2k design that can be run at maximum clock speed.

A. Ignoring hazards, how should the pipeline be modified to **minimize** stalls of Pacer given the limitations of the caches? You cannot add any new memories/caches or increase the number of read/write ports. List all your modifications below the diagram – note you may *not* need to fill in all entries. [10 pts]



Note – the intent of this problem was to not make a superscalar pipeline – thus no new memory ports, register file ports, or ALUs should be added. The objective is to maintain as close to a CPI of 1 as possible with the restricted memories.

Change 1: Extend IF/ID to hold 2 instructions (I1/I2), enable writes every other cycle

Change 2: Add MUX (or MUXes) to Decode to select between I1 and I2 on alternating cycles

Change 3: Compute PC+2 rather than PC+1 in the Fetch Stage

Change 4: Extend MEM to 2 stages by added MEM1/MEM2 latch in the middle. LW/SW now take 2 cycles, so you want to pad out the execution of all other operations so there are no collisions at WB.

Change 5: Optional – Add extra set of address/data registers to data memory, and logic to check for consecutive addresses to enable LW/SW's of 2 words rather than 1. By default the data memory/cache just operates sub-optimally doing 1 LW or SW every 2 cycles.

- Problem 9 continued -

B. Assuming that branches are handled using speculate and squash, and are still resolved in the MEM stage, answer the following about the Pacer design: [4 pts]

- What is the average branch penalty for a mispredicted branch in Pacer? 4 cycles
Nops must be inserted into EX/MEM, ID/EX, and both slots in IF/ID
- To support predict not-taken, are any further changes from part A needed to the pipeline. Briefly explain.

No changes are required for predict not-taken. The instruction memory/cache automatically fetches PC and PC+1, so branches are naturally predicted fall through regardless of which of the 2 (or both) instructions are branches. One small change is converting both fetched instructions in IF/ID to NOPs.

C. What (if any) changes to data forwarding are needed in your design? Are there any additional hazards in your design compared to the baseline LC2k design? Briefly explain. [7 pts]

Data forwarding – additional data forwarding connections must be added for the new MEM stage that was inserted, so between MEM1/MEM2 and EX.

Other hazards – First, the LW-USE distance is increased to 2, so 2 NOPs need to be inserted. Second, there is a structural hazard with doing back-to-back memory operations. So, a NOP needs to be inserted between consecutive LW/SWs. Note, if you propose to combine consecutive LW/SW in your modified MEM stage, then the NOP only needs to be inserted when the LW/SW addresses are not consecutive.

10. Caches I [15 pts]

Consider a cache with the following configuration:

Cache size: 8 bytes

Block size: 2 bytes

Associativity: 2-way

Write Allocate

LRU replacement policy

Address size: 16 bits, byte-addressable

- A. Given the following address, state the tag, set index, and block offset in **binary** form [3 pts]:

0xFEED = 1111_1110_1110_1101 in binary

- Tag = 1111_1110_1110_11
- Set Index = 0
- Block Offset = 1

- B. Given the following series of accesses, specify Hit or Miss, and for misses classify them as Compulsory, Conflict, and Capacity. [12 pts]

Address	3	1	4	9	1	11	6	4	12	0
Type	Comp	Comp	Comp	Comp	Conf	Comp	Comp	Cap	Comp	Cap

11. Caches II [15 pts]

Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to determine the configuration of the cache. Assume 12-bit addresses. **Justify your answers – Guesses will be given zero credit.**

0x310 - Miss
0x30f - Miss
0x510 - Miss
0x31f - Hit
0x72d - Miss
0x72f - Hit
0x320 - Miss
0x520 - Miss
0x720 - Miss

Block size	16 bytes
Associativity	2-way
Number of sets	16

Block Size:

0x31f is a hit. This is possible only if 0x31f is in the same cache block as any one of the preceding 3 accesses (spatial locality). 0x310 is the closest reference to 0x31f. For address locations 0x31f and 0x310 to be a part of the same cache block, the cache block size has to be at least 16 bytes ($0x31f - 0x310 + 1 = 16$ bytes). Also, the block size cannot be greater than 16 bytes, because otherwise 0x30f would have been a hit as well.

Associativity:

Now that we know block size is 16 bytes, we can infer that 0x720 should be a part of the same cache block as 0x72d and 0x72f. But access to 0x720 is a miss. This is possible only if 0x320 *and/or* 0x520 maps to the same set as 0x720. Since two cache block accesses was sufficient to evict 0x720's cache block, cache set's associativity cannot be greater than 2. Also, it cannot be a direct mapped cache, because otherwise 0x510 would have evicted the cache block 0x310-0x31f and access to 0x31f would have been a miss.

$$\begin{aligned}\text{Number of sets} &= \text{Cache size} / (\text{associativity} * \text{block-size}) \\ &= 512 \text{ bytes} / (2 * 16 \text{ bytes}) = 16\end{aligned}$$