CDA 3101 # Midterm Exam #3 **Fall 2013**

PRINT YOUR NAME: _____KEY_____ UFID [5:8]: _____

RECITATION DAY/PERIOD: _____ CDA3101 NUMBER: _____

I have not looked at anyone else's paper, and I have not obtained unauthorized help in completing this exam. Also, I have adhered to and upheld all standards of honesty as stated in the University Honesty Policy and in the course syllabus.

YOUR SIGNATURE: _____ DATE: _____

YOU ARE ALLOWED TO HAVE ONLY THE FOLLOWING ON YOUR DESK OR WORKTABLE:

1. **A *HANDWRITTEN* CRIB SHEET 8-1/2 x 11 inches (BOTH SIDES USABLE)**
2. **A BASIC CALCULATOR – NO GRAPHING, NO PROGRAMMABILITY**
3. **YOUR PEN OR PENCIL AND AN ERASER**

ALL OTHER OF YOUR BELONGINGS BROUGHT INTO THE CLASSROOM MUST BE ON THE FLOOR, UNLESS YOU REQUIRE EYEGLASSES OR AN OPTICAL MAGNIFYING GLASS.

PLEASE TURN ALL CELL PHONES, iPHONES, AND ELECTRONIC DEVICES (EXCEPT CALCULATORS), OFF AND PUT THEM AWAY – RINGTONES AND CELL PHONE CONVERSATIONS ARE NOT PERMITTED DURING EXAM.

CDA 3101 # Midterm Exam #3 **Fall 2013**

PRINT YOUR NAME: _____KEY_____ CDA3101 NUMBER: _____

SCORES: TOTAL EXAM SCORE: ____ / **160** pts ***This part for Instructor & TAs only.***

Q1: ____ Q2: ____ Q3: ____ Q4: ____ Q5: ____ EC: ____

This exam has five regular questions and one extra-credit question. Complete questions that are easiest for you first, then complete what you can of the difficult questions. There is no penalty for guessing. However, *on questions involving calculation, you must show your work. If you do not show your work, you risk getting only partial credit for any answer*.

Q1. (25 pts) **Define these terms using 1-3 sentences, a formula, or diagram:** **(5 pts each)**

(a)Forwarding (in a pipeline processor) – Pipeline forwarding involves routine of data (called *packet* or *packets*) from one pipeline stage to another. In particular, PF is optimal from various points of view:

1. *High efficiency in utilization of network resources*, which enables accommodating a larger amount of traffic on the network, thus lowering operation cost and being the foundation for accommodating the exponential growth of modern networks.
2. *Low implementation complexity*, which enables the realization of larger and more powerful networking systems at low cost, thus offering further support to network growth.
3. *High scalability*, which is an immediate consequence of the above two features.
4. *Deterministic and predictable operation* with minimum delay and no packet loss even under full load condition, which is key in supporting the demanding requirements of the new and valuable services that are being deployed, or envisioned to be deployed, on modern networks, such as telephony, videoconferencing, virtual presence, video on demand, distributed gaming.

Source: http://en.wikipedia.org/wiki/Pipeline_forwarding

(b) Data Hazard (in a pipeline processor) Data hazards occur when instructions that exhibit data dependence modify data in different stages of a pipeline.

Source: http://en.wikipedia.org/wiki/Hazard_(computer_architecture)#Data_hazards

(c) **Control Hazard** (in a pipeline processor) Branching hazards (also known as control hazards) occur with branches. On many instruction pipeline microarchitectures, the processor will not know the outcome of the branch when it needs to insert a new instruction into the pipeline (normally the *fetch* stage). To avoid control hazards microarchitectures can:

- insert a *pipeline bubble* (discussed above), guaranteed to increase latency, or

- use branch prediction and essentially make educated guesses about which instructions to insert, in which case a *pipeline bubble* will only be needed in the case of an incorrect prediction

In the event that a branch causes a pipeline bubble after incorrect instructions have entered the pipeline, care must be taken to prevent any of the wrongly-loaded instructions from having any effect on the processor state excluding energy wasted processing them before they were discovered to be loaded incorrectly.
Source:
http://en.wikipedia.org/wiki/Hazard_(computer_architecture)#Control_hazards_.28branch_hazards.29

(d) Structural Hazard (in a pipeline processor) A structural hazard occurs when a part of the processor's hardware is needed by two or more instructions at the same time. A canonical example is a single memory unit that is accessed both in the fetch stage where an instruction is retrieved from memory, and the memory stage where data is written and/or read from memory.[3] They can often be resolved by separating the component into orthogonal units (such as separate caches) or bubbling the pipeline.
Source: http://en.wikipedia.org/wiki/Hazard_(computer_architecture)#Structural_hazards

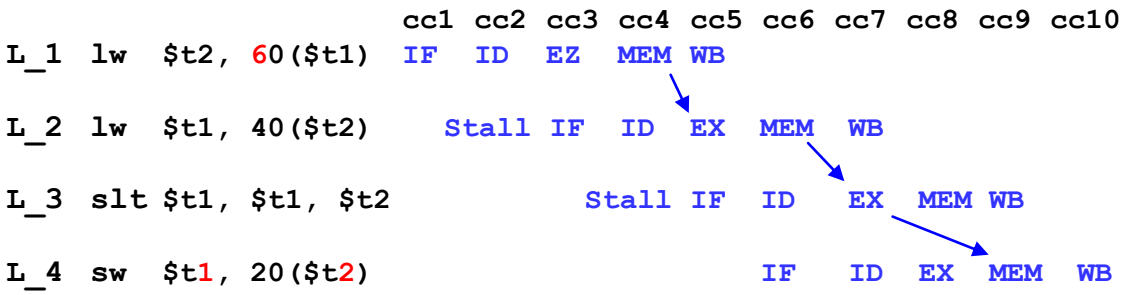
(e) Branch Delay Slot When a branch instruction is involved, the location of the following delay slot instruction in the pipeline may be called a **branch delay slot**. Branch delay slots are found mainly in DSP architectures and older RISC architectures. MIPS, PA-RISC, ETRAX CRIS, SuperH, and SPARC are RISC architectures that each have a single branch delay slot; PowerPC, ARM, and the more recently designed Alpha do not have any.
Source: http://en.wikipedia.org/wiki/Delay_slot

Q2. (30 pts) **The following MIPS program is to be run on a MIPS pipeline processor of form IF-ID-EX-MEM-WB.**

2.1) Please identify all data dependencies beside each instruction, in the form: <type> on <register> from <line-no> to <line-no>, as *WAW on $t6 from L_8 to L_10* (10 pts)

`L_1 lw $t2, 60($t1)`	No dependencies b/c no previous instructions	
`L_2 lw $t1, 40($t2)`	WAR on t1 re: L_1,	RAW on t2 re: L_1
`L_3 slt $t1, $t1, $t2`	WAR on t1 re: L_1, WAW on t1 re: L_2, RAW on t1 re: L_2	RAW on t2 re: L_1 RAR on t2 re: L_2 (not required)
`L_4 sw $t1, 20($t2)`	RAW on t2 re: L_1 RAR on t2 re: L_2 (not required) RAR on t2 re: L_2 (not required)	RAW on t1 re: L_2 and L_3

2.2) Work out and diagram the *optimal* pipeline schedule using forwarding from EX or MEM stages to any other stage, then compute the pipeline CPI: (20 pts)



CPI = _10__ cycles / _4__ instructions = __2.5 cycles per instruction___

Q3. (30 pts) **Answer the following questions with 1-3 SENTENCES to get full credit:**

3.1) What is a multicycle datapath, and what does it do?

Multi-cycle implementations break up instructions into separate steps:
Each step takes a single clock cycle
Each functional unit can be used more than once in an instruction, as long as it is used in different clock cycles
Reduces amount of hardware needed
Reduces average instruction time

Source: http://www.cs.umd.edu/class/fall2003/cmsc311/Lectures/lecture33/multi_cycle.pdf

3.2) What are the advantages and disadvantages of a single-cycle datapath?

Single-cycle datapath implementations execute each instruction in a single clock cycle, which can be slow due to the longer settling time of the circuitry (when compared with a multi-cycle datapath).

3.3) What is microcoding and why was it developed?

Microcode is a layer of hardware-level instructions or data structures involved in the implementation of higher level machine code instructions in central processing units, and in the implementation of the internal logic of many channel controllers, disk controllers, network interface controllers, network processors, graphics processing units, and other hardware. It resides in special high-speed memory and translates machine instructions into sequences of detailed circuit-level operations. It helps separate the machine instructions from the underlying electronics so that instructions can be designed and altered more freely. It also makes it feasible to build complex multi-step instructions while still reducing the complexity of the electronic circuitry compared to other methods.

Source: http://en.wikipedia.org/wiki/Microcode

CDA 3101 **Midterm Exam #3** Fall 2013

PRINT YOUR NAME: _____KEY_____ CDA3101 NUMBER: _____

Q4. (50 pts) **The following MIPS program is to be run on a MIPS pipeline processor of form IF-ID-EX-MEM-WB. *You may re-order independent instructions, but correctly.***

4.1) Please identify all data dependencies beside each instruction, in the form: <type> on <register> from <line-no> to <line-no>, as *WAW on $t6 from L_10 to L_8* (20 pts)

```
L_1  sub $t2, $t1, $t3
```
No dependencies b/c no previous instructions

```
L_2  slt $t4, $t5, $t4
```
No dependencies with respect to L_1

```
L_3  beq $t4, $zero, BTA
```
("BTA" is somewhere else)

RAW on t4 re: L_2

```
L_4  lw  $t1, 80($t5)
```
WAR on t1 re: L_1 RAR on t5 re: L_2 (not req'd)

4.2) Work out and diagram the *optimal* pipeline schedule using forwarding from EX or MEM stages to any other stage, then compute the pipeline CPI: (30 pts)

```
                          cc1  cc2  cc3  cc4  cc5  cc6  cc7  cc8  cc9  cc10
L_1  sub $t2, $t1, $t3    IF   ID   EX   MEM  WB
L_2  slt $t4, $t5, $t4         IF   ID   EX   MEM  WB
L_3  beq $t4, $zero, BTA       IF   ID   EX   MEM  WB
     ("BTA" is elsewhere)
L_4  lw  $t1, 80($t5)               IF   ID   EX   MEM  WB   [branch not taken]
```

CPI = __8__ cycles / __4__ instructions = __2 cyc/instr___ [assume branch not taken]

Q5. (25 pts) **Given a 4-way set associative cache of total size 2MB that has a 26-bit cache address and blocks of size 8KB each, answer the following questions.**
You must show all work for full credit. (8 pts each + 1 free point)

5.1) How many bits in the "index" field of the cache address? Justify your answer.
If the cache is 4-way set associative, then the number of sets is not 4. Rather, the number of sets is the number of rows in the cache divided by 4. Since this cache has 256 rows (2MB / 8KB), there are 64 sets. The answer is then: log_2(64) = 6 index bits

5.2) How many bits in the "offset" field of the cache address? Justify your answer.
Observe: 8KB/block ➔ 2^{13}bytes per block ➔ $\log_2(2^{13})$ = 13 bits if byte-aligned
Assuming that the blocks are word-aligned, then we have Offset_bits = 13bits–2 bits = 11bits

5.3) Diagram the cache address structure with tag, index, and offset fields, and show how many bits are assigned to each field: (9 bits + 6 bits + 11 bits = 26 bits)

Tag = 7 bits	Index = 6 bits	Offset = 13 bits

Note: "7 bit tag / 6 bit index / 13 bit offset" is acceptable as an answer for students who assumed byte-aligned caches.

Note: "11 bit tag / 2 bit index / 13 bit offset" is acceptable as an answer for students that assumed byte-aligned caches.

Extra Credit Problem: (20 pts) **A pipeline with 18 stages runs a program *P* having 4,076 instructions. Branches comprise 17 percent of the instructions, and the "branch not taken" assumption holds for branch prediction. Further assume that 41 percent of the branches are predicted correctly, and there is an average penalty of 1.7 cycles for each mispredicted branch. Additionally, 2 percent of the total instructions incur an average of 1.3 stalls each.**

CALCULATE THE CPI OF *P* ON THIS PIPELINE – SHOW ALL WORK TO GET FULL CREDIT IF YOUR ANSWER IS CORRECT, PARTIAL CREDIT IF NOT.

Model Variables: N = 18 stages, M = 4,076 instructions, f_{br} = 0.17, f_{be} = 1 – 0.41 = 0.59

L = 1.7 cycle penalty for mispredicted branch,

K = 1.3 stall penalty, f_{stall} = 0.02

Given: $CPI = N_{cyc}/M = 1 + (f_{stall} \cdot K) + (N - 1)/M$ [effect of stalls, Web Sec 5.4]

$CPI = N_{cyc}/M = 1 + [(f_{br} \cdot f_{be} \cdot (L-1)) + (N - 1)/M]$ [effect of branch, Web Sec 5.4]

Step 1: Combine penalty terms from above equations to yield net performance equation:

$$CPI = 1 + (N - 1)/M + (f_{stall} \cdot K) + (f_{br} \cdot f_{be} \cdot (L-1))$$

Step 2: Plug the values from the model variables into the equation from Step 1, to get answer:

$$CPI = 1 + (N - 1)/M + (f_{stall} \cdot K) + (f_{br} \cdot f_{be} \cdot (L-1))$$
$$= 1 + (18-1)/4{,}076 + (0.02 \cdot 1.3) + (0.17 \cdot 0.59 \cdot (1.7 - 1))$$
$$= 1 + (17/4{,}076) + 0.026 + (0.1003 \cdot 0.7)$$
$$= 1 + 0.00417 + 0.026 + 0.07021$$
$$= \mathbf{1.10038}$$