Midterm Exam #2

Prof. Matthew O'Keefe
TA: Eric Sepannen

Department of Electrical and Computer Engineering

University of Minnesota
Twin Cities Campus

EE 4363 Introduction to Microprocessors

15 April 2009

Answer all questions. There is a total of 100 points and you should have 12 pages in your test. Please place your name, ID number and recitation number in the space below. *Please:* do it now.

Reminder: This test is closed book and closed notes; no calculators are allowed. Show ALL your work. Additional pages are provided at the end of the exam.

Name:

Student ID Number:

Problem	**Score**
1	(20)
2	(20)
3	(20)
4	(20)
5	(20)
Total	(100)

[1] (20 points) Answer the following questions regarding memory hierarchies.

(a) What are the typical access time ranges for SRAM, DRAM, and magnetic disk technology.

SRAM: 2 -10ns
DRAM: 50-80ns
Magnetic Disk: 5-15 milliseconds

(b) What are the two types of locality in program instruction and data accesses? How can locality be used to exploit a memory hierarchy?

Spatial and temporal locality. Assume an instruction address stream i, i+1, i+2, …, i+n is executed. Temporal locality refers to the fact that this set of instructions is likely to be executed again in the near future. Spatial locality refers to the fact that if we execute an instruction at address i, it's quite likely that we will access instructions i+1, i+2, i+3. Informally, temporal locality means that when we access memory at address i, we are likely to address it again in the near future. Spatial locality means that if we access address i, we are likely to also have accessed i-1, i-2, i-3, and so on, as well as i+1, i+2, i+3, and so on.

(c) Define "memory hierarchy" and the related electronic and storage design principles that drive its existence.

A memory hierarchy consists of a series of memory levels, starting with the level closest to the CPU, such that each level is progressively larger in capacity and slower in access. The key design principle is that faster is more expensive, hence faster memory technologies can only be deployed economically with smaller capacities. Electronic design principles allow small memories to be faster because more power can be used per cell to drive faster switching times, and the smaller physical area covered by the smaller memory means that the control and data signals can travel faster across the smaller memory.

[2] (20 points) Pipelining and processor clock cycle times.

Assume that the individual stages of a MIPS datapath implementation have the following latencies (*ps* represents picoseconds):

Instruction Fetch	Instruction Decode	Execute (ALU)	Memory Access	Register Write Back
200ps	120ps	190ps	400ps	100ps

(a) What is the clock cycle time in a pipelined and non-pipelined implementation version of this MIPS processor?

Pipelined: cycle time determined by slowest stage: 400ps.
Non-pipelined: cycle time determined by sum of all stages: 1010ps.

(b) What is the total latency of the `lw` instruction in a pipelined and non-pipelined processor?

LW instruction uses all 5 stages.
Pipelined processor takes 5 cycles at 400ps per cycle for total latency of 2000ps.
Non-pipelined processor takes 200+120+190+400+100 = 1010ps.

(c) 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?

Split Memory Access stage into two stages of 200ps. New clock cycle time is 200ps.

[3] (20 points) Cache access patterns.

Assume a fixed-size processor cache with two alternative configurations (direct-mapped and 2-way set associative), each with total size of 4 blocks, where each cache block is 16-bytes (4 32-bit MIPS words) wide.

(a) What is the hit rate for each of the 2 cache configurations if the byte address stream for memory access is 0, 16, 32, 48, 64…? What is the miss rate for this byte address stream?

Direct-mapped cache:

Address:	*0*	*16*	*32*	*48*	*64…*
Block #/ set #:	*0/0*	*1/1*	*2/2*	*3/3*	*0/0…*
H or M:	*M*	*M*	*M*	*M*	*M…*

Hit rate: 0%; Miss rate is 1.0-(Hit rate)=100%

2-way-set-associative cache (set number is block number modulo (#sets=2):

Address:	*0*	*16*	*32*	*48*	*64…*
Block #/ set #:	*0/0*	*1/1*	*2/0*	*3/1*	*0/0…*
H or M:	*M*	*M*	*M*	*M*	*M…*

Hit rate: 0%; Miss rate is 1.0-(Hit rate)=100%

(b) What is the hit rate for each of the two cache configurations if the byte address stream is 0, 4, 8, 12, 16, 20, 24, 30…?

Direct-mapped cache:

Address:	*0*	*4*	*8*	*12*	*16*	*20*	*24*	*28*	*32…*
Block #/ set #:	*0/0*	*0/0*	*0/0*	*0/0*	*1/1*	*1/1*	*1/1*	*1/1*	*2/2*
H or M:	*M*	*H*	*H*	*H*	*M*	*H*	*H*	*H*	*M*

Hit rate: 75%; Miss rate is 1.0-(Hit rate)=25%

2-way-set-associative cache (set number is block number modulo (#sets=2):

Address:	*0*	*4*	*8*	*12*	*16*	*20*	*24*	*28*	*32…*
Block #/ set #:	*0/0*	*0/0*	*0/0*	*0/0*	*1/1*	*1/1*	*1/1*	*1/1*	*2/0*
H or M:	*M*	*H*	*H*	*H*	*M*	*H*	*H*	*H*	*M*

Hit rate: 75%; Miss rate is 1.0-(Hit rate)=25%

(c) What is the hit rate for each of the two cache configurations if the byte address stream is 0, 64, 32, 96, 0, 64, 32, 96…?

Direct-mapped cache:

Address:	*0*	*64*	*32*	*96*	*0*	*64*	*32*	*96*
Block #/ set #:	*0/0*	*4/0*	*2/2*	*6/2*	*0/0*	*4/0*	*2/2*	*6/2*
H or M:	*M*	*M*	*M*	*M*	*M*	*M*	*M*	*M*

Hit rate: 0%; Miss rate is 1.0-(Hit rate)=100%

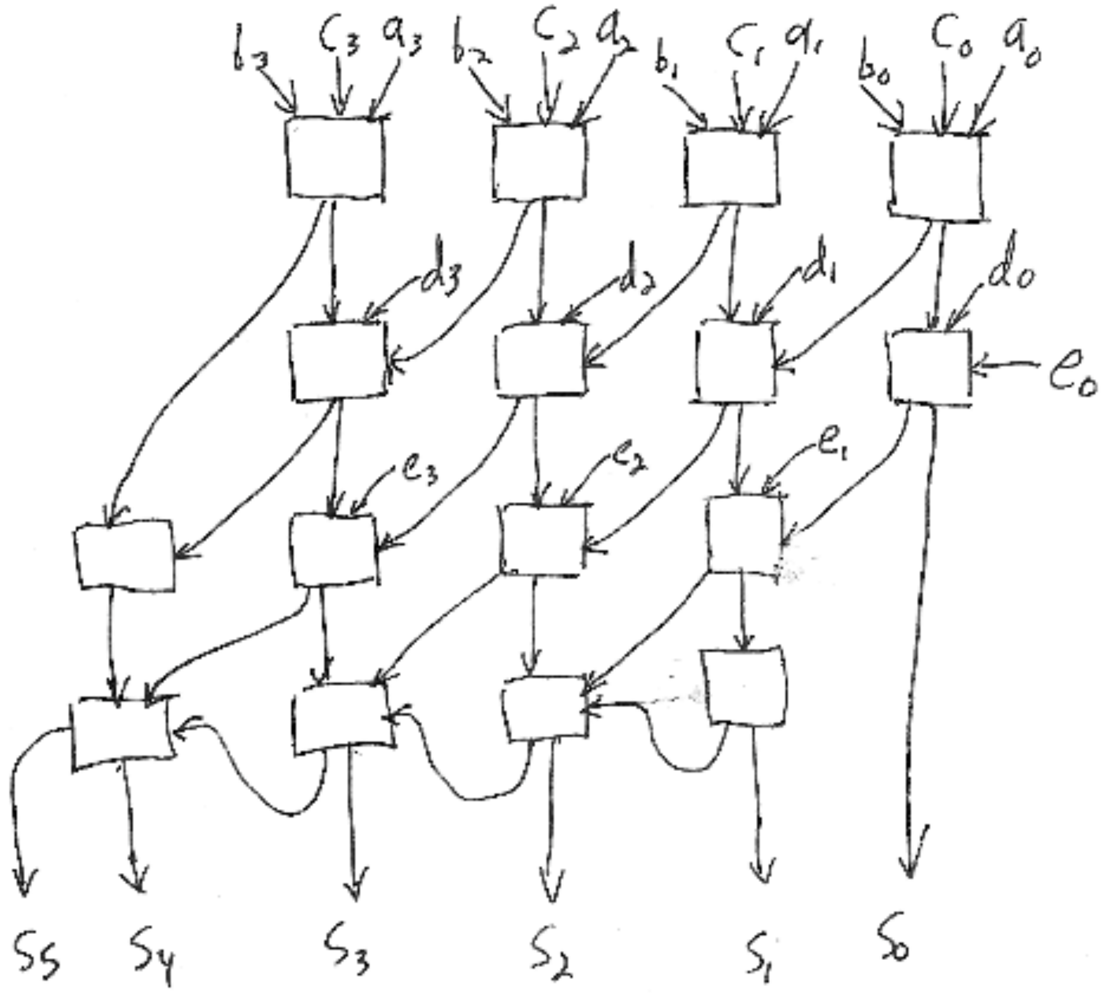
2-way-set-associative cache (set number is block number modulo (#sets=2):

Address:	*0*	*64*	*32*	*96*	*0*	*64*	*32*	*96*
Block #/ set #:	*0/0*	*4/0*	*2/0*	*6/0*	*0/0*	*4/0*	*2/0*	*6/0*
H or M:	*M*	*M*	*M*	*M*	*M*	*M*	*M*	*M*

Hit rate: 0%; Miss rate is 1.0-(Hit rate)=100%

[4] (20 points)

Show an efficient adder design for adding 5 4-bit unsigned numbers using 3 rows of carry save adders and a final row with a traditional adder. If each adder unit has a time delay *t,* what is the length of the critical path in a non-pipelined design? [*7t*] What is the length of the critical path in a pipelined design with pipeline registers between each row and using carry-propagate techniques in the final row? [5t]



[5] (20 points) Reordering Code to Avoid Pipeline Stalls.

Consider the MIPS pipeline implementation shown on the next page, which is consistent with the basic MIPS pipeline from sections 4.5 and 4.6 in the text and discussed in class. Assume this pipeline has implemented all necessary hazard detection and forwarding to remove stalls and delay wherever possible

For the following C fragment:

```
A = B + C;
D = A + E;
```

the following sequence of MIPS instructions can be executed (assume all variables are in memory and are addressable as offsets from `$t0`):

```
lw   $t1, 0($t0)    ; load B
lw   $t2, 4($t0)    ; load C
add  $t3, $t1, $t2  ; add B and C, result to A
sw   $t3, 8($t0)    ; store A
lw   $t4, 12($t0)   ; load E
add  $t5, $t3, $t4  ; add A and E, result to D
sw   $t5, 16($t0)   ; store D
```

(a) Indicate the read-after-write dependencies for the MIPS instruction sequence shown. Indicate where pipeline stalls will occur.

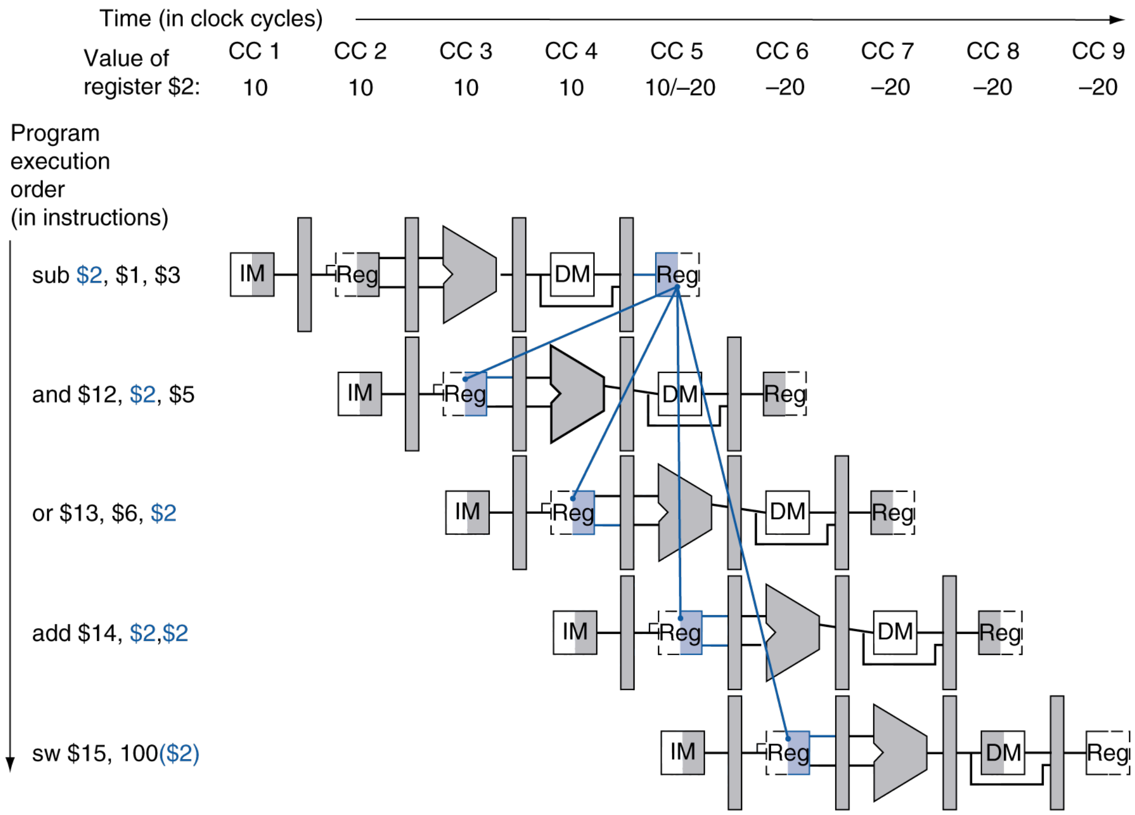
RAW from 1-3 ($t1), 2-3 ($t2), 3-4($t3), 5-6($t4), 6-7($t5)
one-cycle stall during 3 (waiting for $t2), 6 (waiting for $t4).

(b) Re-order the instruction sequence shown above to remove any pipeline stalls that occur. What is the reduction in clock cycles required between your re-ordered instruction sequence and the original sequence?

```
lw   $t1, 0($t0)    ; load B
lw   $t2, 4($t0)    ; load C
lw   $t4, 12($t0)   ; load E
add  $t3, $t1, $t2  ; add B and C, result to A
sw   $t3, 8($t0)    ; store A
add  $t5, $t3, $t4  ; add A and E, result to D
sw   $t5, 16($t0)   ; store D
```

1, 2, 5, 3, 4, 6, 7: 2 fewer cycles (both stalls eliminated).

Note: The instruction sequence shown is unrelated to the instruction sequence described in the problem. It is shown here for information purposes only.



This page deliberately left empty.