Home Work 2
ECE3056
Fall 2014

This homework is to be turned in online on T-square. Handwritten (scanned) solutions are fine too.

You will be graded on your attempt to solve the problem, and not on correctness. If you attempt more than half of the questions and submit the solutions before the deadline, you will get the full 2 points regardless of your answers.

The solutions to the homework will be posted shortly after the submission deadline. The objective is that you will compare your solutions with ours and determine where you could have gone wrong.

Remember, even though this homework is only 2 points, if you do it with seriousness it will help you identify where you may be weak, and thus help you get more points on the Midterm (which is worth 20 points!).

The first few pages of the homework are taken from the midterm from my last offering of ECE3056.

Good Luck!

Department of Electrical and Computer Engineering
Georgia Institute of Technology

ECE3056A: Architecture, Concurrency, and Energy
Moinuddin K. Qureshi, Instructor

Exam 2, April 9, 2013

Name : Hongyao Shi

GT Account: hshi38

Problem 1 (20 points): _____

Problem 2 (20 points): _____

Problem 3 (20 points): _____

Problem 4 (20 points): _____

Problem 5 (20 points): _____

Total (100 points) : _____

This exam is given under the Georgia Tech Honor Code System. Anyone found to have submitted copied work instead of original work will be dealt with in full accordance with Institute policies.

Georgia Tech Honor Pledge: "**I have neither given nor received aid on this exam."**

[**MUST sign**:] _____

Note: Where needed, show all your intermediate results to receive full credit. Do all your work in this examination handout. Use the back of the exam sheets if necessary.

Note: For your benefit, we have provided the ISA instructions, datapath, State machine, and control store table for LC-3b at the end of the exam sheet.

GOOD LUCK!

Name:_____

Problem 1 – "Potpourri" (20 points, there are four parts, each worth 5 points)

I. Write-back and Write-through are two policies for a cache.
Which policy require larger tag-store overhead? write back Why (less than five words)?

Need dirty bit each line

The disadvantage of write-through policy is that it requires higher write time

II. Interrupts and exceptions are similar in that they both stop an executing process to execute a very different segment of code. However, two primary differences between interrupts and exceptions are as follows (in less than ten words each):

1. Exceptions caused internally while interrupts caused externally to running thread

2. Exceptions handled when detected, interrupts handled when convenient

III. We have a 16KB L1 cache with linesize of 128 bytes. We wish to operate this cache without having TLB in the critical path of cache access while incurring very little hardware and software complexity. To do so, we must architect the cache to be (four words only) virtually indexed physically tagged

IV. We have a machine with 36-bit address space. We wish to use this machine with an OS that uses a pagesize of 4KB. If each Page Table Entry (PTE) is 4-bytes, the size of the page table will be 64 MB

Name:_____

Problem 2 "Power of Vector Processors" (20 points) –

Consider the following code that averages four values, operating on arrays of 64 elements.

```
for(i=0; i<64; i++){
   A[i]=(B[i]+C[i]+D[i]+E[i])/4
}
```

Assume that memory accesses take 11-cycle for an operation (Load or Store), ADD takes 2 cycles, and SHF operation takes 1 cycle.

I. Write the above code for a single-issue in-order scalar machine that contains auto-increment memory addressing mode and a single instruction (DECBNZ) that can do decrement of register value and conditional branch in one cycle.

```
            add r0 #63
loop:       ldw r1 B
            ldw r2 C
            ldw r3 D
            ldw r4 E
            add r5 r1
            add r5 r2
            add r5 r3
            add r5 r4
            rshfa r5 2
            stw r5 A
            DECBNZ   r0  loop
```

How many dynamic instructions does this code execute: 705

How many cycles will this code take to execute: 4354

NOTE: We did not cover Vector Processors yet. So, the portion on Vector Processors is removed from this question.

Name: _____

Problem 3 “Architecting New Instruction” (20 points)

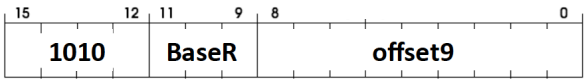
We want to add a new instruction ByteSwap (BSWAP) to the LC-3b ISA using the unused opcode “1010”. The specification of the BSWAP instruction is shown below:

BSWAP

Assembler Formats

BSWAP BaseR, offset9

Encoding



Operation

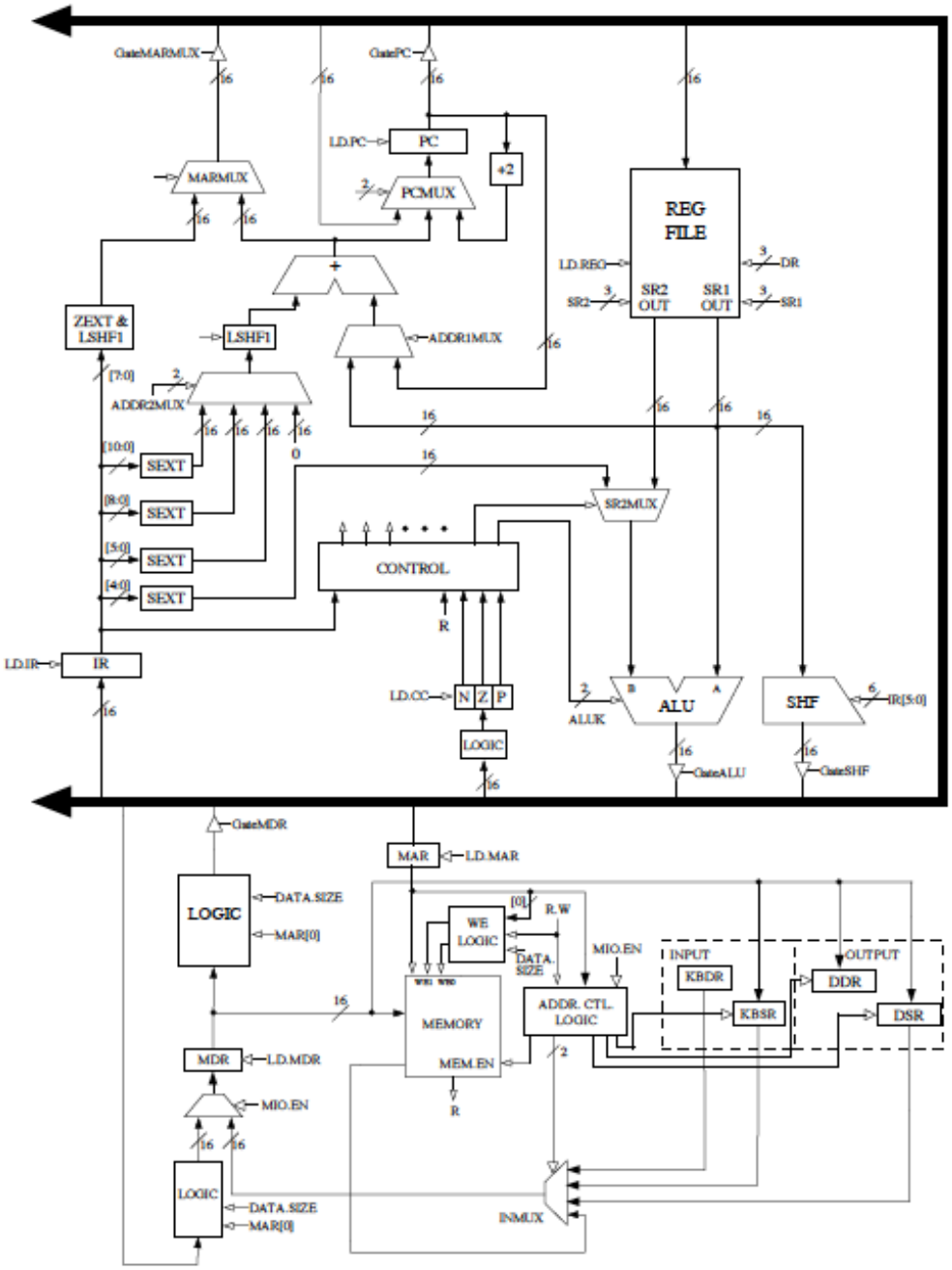
First Byte: MEM[BaseR + LSHF(SEXT(offset6), 1)]
Second Byte: MEM[BaseR + LSHF(SEXT(offset6), 1)+1]

Swap Contents of (First Byte, Second Byte)

A word aligned memory location is accessed. The address of this location is computed with a Base Register and a 9-bit word offset. The two bytes of the word are swapped. This modified value is written to the memory word. Note, none of the register values or condition codes should be affected by the execution of this instruction (except for PC=PC+2, of course).

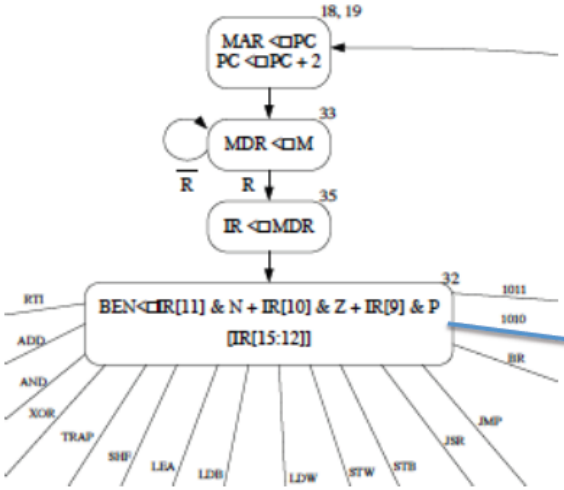
I. Show the changes to the datapath of LC-3b as shown on the next page. While you are free to change the datapath any way you like, you should try to keep the changes to a minimum (simple solution is desirable).

Problem 3 Continued



Problem 3 Continued

II) Your job now is to develop the state assignments for the BSWAP instruction, and define what action needs to happen in each state. You must also specify the **control signals** that must be asserted in each newly added state (ignore the control signals that are not asserted). If your newly added structures to the datapath require any control signals, you must specify those signals as well. You can add as many states as you need. You must also specify the cond bits, J-bits, and state numbers (feel free to use any unused states, but your state assignments should work with the existing micro-sequencer). The last state must be connected back to state 18/19.



10

BaseR=BaseR+LSHF(Offset9,1)

11

STW R7 , x4000

50

Load word into R7 from Memory in Base R

51

STB R7 BaseR #0

52

right shift R7 by 8 bits

53

STB R7 BaseR #1

54

LDW R7 x4000

Problem 3 Continued

III) Assuming memory access takes 1 cycle, how many cycles would it take to execute the BSWAP instruction?

11

IV) If we did not have the BSWAP instruction, how would you perform the BSWAP operation using existing instructions?

See Question 2

V) Assuming memory access takes 1 cycle, how many cycles would it take to execute the set of instructions in (IV)?

44

Name: _____

Problem 4 “Caching Insights” (20 points)

An LC-3b computer system has a 256 byte physically addressed cache, with a linesize of 16 bytes.

The following loop is executed on this machine:

```
sum=0;

for(i=0; i<256; i++){
        sum +=  (Y[i]*Z[i]);
}
```

The starting address of the array are:

X: x4000
Y: x5180
Z: x6280

Each element in the each array is stored as a 16-bit word. Assume that **sum** and **i** stored in registers.

I) Calculate the cache hit ratio for the loop if the cache is direct mapped. __0%___

II) What is the minimum associativity of the cache that will yield an improvement in hit rate, assuming that the cache capacity remains at 256B? Please be brief and specific

2 columns for each address since their modulation land at same positions

III) What is the improved hit rate with the new cache configuration? __93.8%__

Name: _____

Problem 5 "DRAM Systems" (20 points)

We have a DRAM-based memory system of 64KB. This system has row-buffer size of 256 bytes, and 16-banks. The data-mapping policy is geared towards increasing the hit rate for applications with good spatial locality.

To open a row it takes 10 ns, to read/write from an open row it takes 8 ns, and to precharge (close) the row it takes 10ns. Assume that the row buffer of associated with each banks is initially empty.

Assume that the system has a cache linesize of 16 bytes. Shown below is a stream of eight cache misses that arrive at the memory system (note that the address that arrives at the memory system is the line-address and not the byte address).

0xCAB, 0xCAD, 0xBAD, 0xDAB, 0xBBB, 0xCCC, 0x100, 0x10F

You can assume that a request arrives at the memory system a long time after a previous request has been satisfied. The memory system supports both open-page and closed-page policies.

I) If we employ an open page policy, specify whether the row-buffer access outcome is a row buffer hit, row buffer miss, or row buffer conflict. Also specify the time required to service each access. Given your data mapping policy you should also specify the ID of the bank that a given request maps to.

Accesss	0xCAB	0xCAD	0xBAD	0XDAB	0xBBB	0xCCC	0x100	0x10F
BankID	A	A	A	A	B	C	0	0
Rowbuf access Outcome	empty	hit	miss	miss	empty	empty	empty	hit
Latency (ns)	18	8	28	28	18	18	18	8

What is the total time to service the eight requests? 144

II) If we employ a closed-page policy, specify the time required to service each access. You may also specify the ID of the row-buffer for the given request, and the ID of the bank that request maps to.

Accesss	0xCAB	0xCAD	0xBAD	0xDAB	0xBBB	0xCCC	0x100	0x10F
Latency (ns)	18	18	18	18	18	18	18	18

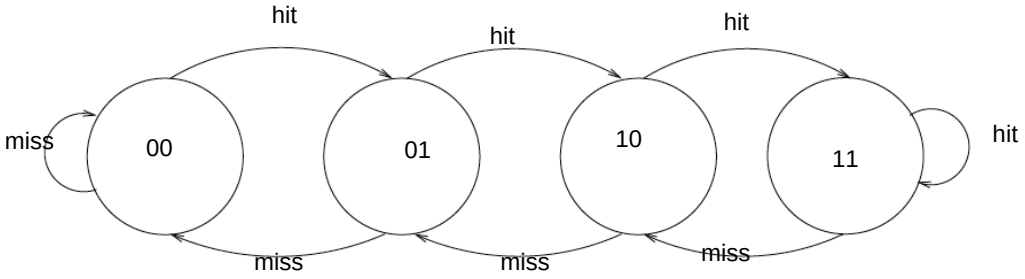
What is the total time to service the eight requests? 144

III) What policy would you recommend, open-page or closed-page? closed page

Problem 6

Part a (5 points): A dynamic branch predictor introduced by Jim Smith at ISCA in 1981 involved the following state diagram. Depending on which state it is in at any particular time, a branch will be predicted as taken or not taken.

Complete the state diagram by (a) labeling each arc, thereby indicating what causes the branch predictor to transfer from one state to another, and (b) writing "Predict taken" or "Predict not taken" inside each state, thereby indicating the branch prediction that will occur when a branch is fetched and decoded and the branch predictor is in each of the four states.



Part B omitted

Part c (7 points): We wish to design a 64 KB physical cache for an ISA having a 4KB page size. We want to keep the TLB out of the critical path, but refuse to add the complexity of a virtually indexed, physically tagged cache. What is the **minimum** associativity of our design that satisfies the above constraints?

Answer: 8

Part d (8 points): A CMP processor consists of 8 simple cores on a single chip. A program consists of code that is 80% parallelizable. What is the speed-up obtained by executing this program on the 8-core chip versus running it on a chip that only has a single simple core.

Answer: 333%

Suppose instead we execute this program on a chip where 4 of the 8 simple cores have been replaced by one heavy-weight core that processes 2.5 times faster than a simple core. What is the speed-up of running on this chip versus running on a chip that has eight simple cores. Assume that when the chip is processing the parallelizable code, the heavyweight core is idle.

Answer: 107%

The below is part E (and not D)

Part d (5 points): To perform a DRAM access, do you always need to assert the Row Address Strobe (RAS) so that the high bits of address are applied to the DRAM chip? **Circle one:** Yes/No. Why or why not? Explain.

No

Q7:

A byte-addressable, write-back cache of fixed total size and fixed line size is implemented as both a direct mapped cache and also as an N-way set-associative cache. In both cases, we will assume the cache is initially empty.

First, consider the cache organized as a direct mapped cache. The following sequence of 11 accesses generates the hits/misses shown.

Address	R/W	Direct Mapped (Miss/Hit)
0100001010	R	miss
1100100111	R	Miss
1110101000	R	Miss
0011000101	R	miss
0110111100	R	miss
1010110101	R	Miss
1100100000	R	Miss
0100001111	R	Hit
0101111111	W	Miss
0110110100	R	miss
0110100101	R	Miss

Part a: What is the cache line size? 16bytes

Explain why. Please be concise but clear

The hit line compares to the previous lines, the first line is similar except the last 4 bits, thus the line size is 2^4=16 bytes

Part b: What are the number of index bits for the direct mapped cache? 3 bits

Explain why. Please be concise but clear.

There are 4 offset bits, 3 index bits and 3 tag bits

Now consider the cache organized as a N-way set-associative cache.

The total size of the tag store for the N-way set associative cache is 112 bits. Each tag store entry contains, in addition to the tag bits, 10 more bits which include the valid bit, modified bit, LRU bits, snoopy cache bits and other bits whose functions do not affect this problem.

We have expanded the table to show the hit/misses for the same sequence of accesses when the cache is organized as an N-way set-associative cache.

Address	R/W	Direct Mapped (Miss/Hit)	N-way associative (Miss/Hit)
0100001010	R		miss
1100100111	R	Miss	Miss
1110101000	R	Miss	miss
0011000101	R		Miss
0110111100	R		Miss
1010110101	R	Miss	miss
1100100000	R	Miss	hit
0100001111	R	Hit	hit
0101111111	W	Miss	Miss
0110110100	R		Hit
0110100101	R	Miss	miss

Part c: What is N? 2

Explain why. Please be concise but clear.

The first hit fits the second miss, thus it must be at least 2 ways

Part d: What is the number of index bits for the N-Way set associative cache? 2

Explain why. Please be concise but clear

since the ways are halfed there is only 4 rows,
thus requires 2 bits

Part e: Is the cache write allocate? Explain why. Please be concise but clear.

No, hit when write does not cause confliction

Part f: Please complete the table above by filling in "H" or "M" for each of the blank entries.

Q8:

A byte-addressable cache of fixed total size and fixed line size is implemented both as a k-way set associative cache and also as a fully associative cache. Assume perfect LRU replacement and line allocate on a write miss policies. For this problem, please show your work below.

If the two implementations are presented with the same sequence of addresses for byte accesses, they generate hits/misses as shown below:

Address	R/W	k-way associative (Miss/Hit)	Fully associative (Miss/Hit)
011011100	W	Miss	Miss
011001100	R	Miss	Miss
111000111	R	Miss	Miss
011011000	R	Hit	Hit
010101100	R	Miss	Miss
001111111	R	Miss	Miss
011001111	R	Miss	Hit
111110010	R	Miss	Miss
110100111	R	Miss	Miss
111000011	R	Hit	Miss

Part a: How many cache lines are there in the two cache implementations? 8

Part b: The set associative cache is k-way, where k =

- ☐ 1-Way (Direct Mapped)
- ☐ 2-Way
- ☐ 3-Way 3
- ☐ 4-Way
- ☐ 5-Way
- ☐ 6-Way
- ☐ 7-Way
- ☐ 8-Way
- ☐ 9-Way

(check the appropriate box).

Part c: For the k-way cache, how many tag bits: 5 and index bits: 1

Part d: Recall in the preamble, that this cache does an allocate on write miss. Complete the hit/miss behavior table if the cache did not allocate on a write miss.

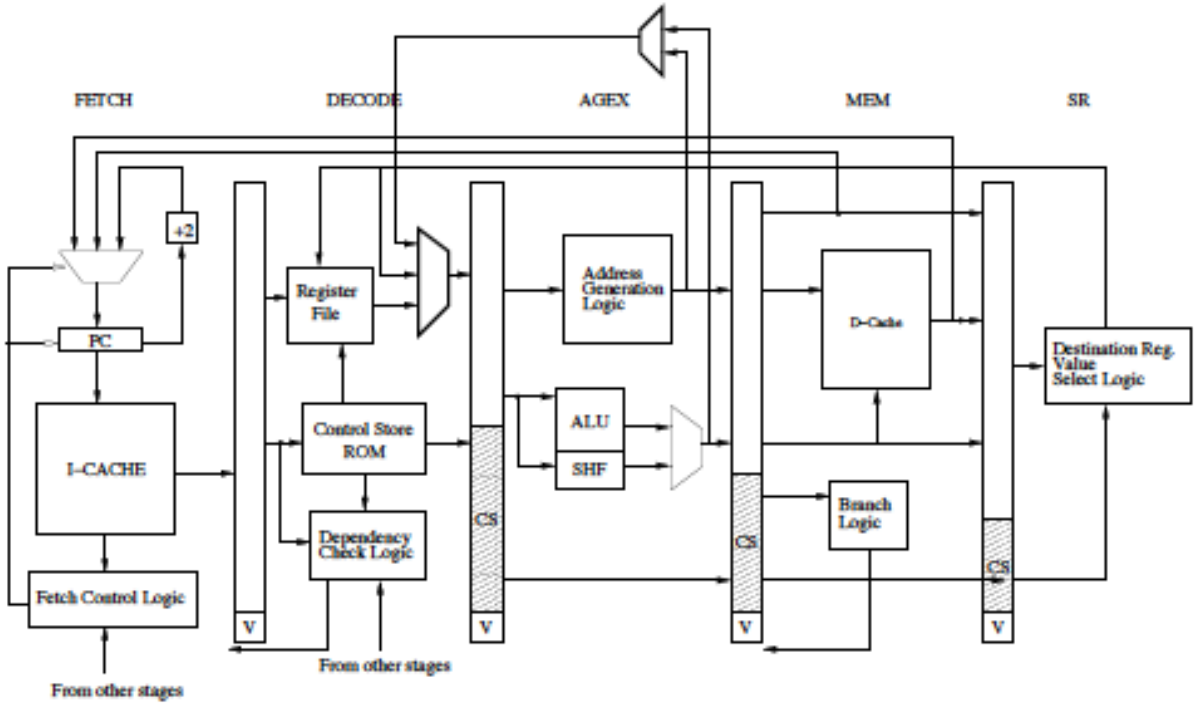
Address	R/W	k-way associative (M/H)	Fully associative (M/H)
011011100	W	Miss	Miss
011001100	R	m	miss
111000111	R	m	miss
011011000	R	m	m
010101100	R	m	m
001111111	R	m	m
011001111	R	m	m
111110010	R	m	h
110100111	R	m	m
111000011	R	H	m

PROBLEM 9:

Recall the pipelined LC-3b implementation from lab6. In addition to Icache and Dcache misses, there are 3 other causes of pipeline stalls. Register dependencies and condition code dependencies are two of them. Assume in this problem that all Icache and Dcache accesses are hits.

Part a (8 points)

First let's consider ONLY stalls caused by register dependencies. We have added forwarding paths to the datapath such that register data is forwarded to the input latches of the AGEX stage from the output of two stages (AGEX and SR) as shown in the figure below. Note that the changes from lab 6 are shown in bold. Assume that all necessary control signals have been added to make this forwarding scheme work. Also, the forwarding logic is implemented for both SR1 and SR2 in the decode stage. We have shown only one mux in the decode stage in the interest of making the diagram clear, rather than cluttered.



LC-3b pipeline diagram with data forwarding

PROBLEM 9 Continued:

Q1) Your job: Provide two different sequences of instructions that will stall in the original LC-3b pipeline from lab 6, but will not stall in the new pipeline with our forwarding scheme.

One instruction sequence must make use of the forwarding path from AGEX to DECODE, and the other instruction sequence must make use of the forwarding path from SR to DECODE. Also, for all instruction sequences, please show which stage of processing each instruction is in (F,D,E,M,SR) during each cycle.

Instructions	cycle 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
add r0 r0 #1	f	d	a	m	s											
add r1 r0 #1		f	d	a	m	s										

Instructions	cycle 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LDW R1 R1 #0	f	d	a	m	s											
add r0 r0 #1		f	d	a	m	s										
add r2 r2 #1			f	d	a	m	s									
add r1 r1 #1				f	d	a	m	s								

Q2) Provide one sequence of 3 instructions using OPERATE instructions ONLY that would cause a stall due to register dependencies even in the new pipeline with our forwarding scheme.

Instructions	cycle 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LDW R1 R1 #0	f	d	a	m	s											
add r0 r0 #1		f	d	a	m	s										
add r1 r1 #1			f	d	d	a	m	s								

What forwarding path can you add to the new pipeline that will remove this stall condition completely.

Answer: Mem to decode forwarding

Q3) By now, we have added all the forwarding paths to the pipeline. Recall the sequence in Q2 was restricted to consist of only operate instructions. Now give us a 2-instruction sequence without this restriction which will still generate a data dependency stall.

Instructions	cycle 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ldw r1 r1 #0	f	d	a	m	s											
add r1 r1 #1		f	d	d	m	s										

PROBLEM 9: Continued …

Part b (8 points)

Now let's consider ONLY stalls due to condition code dependencies.

Q4) Provide a 2-instruction sequence that will stall in the original LC-3b pipeline from lab 6.

Instructions	cycle 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
add r0 r0 #0	f	d	a	m	w											
br np x4000		f	d	d	d	d	a	m	s							

Q5) Suppose we move the CC register from the DE stage to the MEM stage. Branch instructions no longer stall in the DE stage. Instead they keep moving forward, and read the CC register in the MEM stage, and update MEM.PCMUX accordingly. Instructions update the CC register when they are in the SR stage.

With the CC register moved to the MEM stage, how many cycles would your 2-instruction sequence take to execute. Again assume all control signals necessary to make it work are added to the pipeline.

Instructions	cycle 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
add r0 r0 #0	f	d	a	m	w											
br np x4000		f	d	a	m	m	s									

Does the optimization described above completely eliminate stalls related to condition codes dependencies? If yes, tell us why. If no, what can be done to completely eliminate the stalls? Please explain in fewer than 20 words.

No, there is still one more cycle of mem stage needed

Part c (4 points)

Q6) What is the third "other" cause of stalls in the pipelined LC-3b from lab 6, in addition to Icache and Dcache misses?

program flow dependency

How can it be avoided? (fewer than 10 words)

branch prediction

PROBLEM 10:

A processor makes seven successive READ accesses to its physical memory system (PA is 12 bits). The addresses, in random order, are shown below:

x428, x364, x126, x102, x46C, x12B, x265.

Your job is to identify the sequence of the accesses, and fill in the table below with the addresses of each of the seven reads in their correct sequence.

1st	2nd	3rd	4th	5th	6th	7th
102	428	126	12B	265	364	46C

Some information that may be useful:

1. Only the 2nd, 3rd, 5th, 6th, and 7th accesses were seen on the memory bus.

2. The processor has a 512 byte, 2-way set associative, write-through cache with perfect LRU replacement. The line size is 16 bytes.

3. A snapshot of the tag store before the 1st of the seven reads and a snapshot of the tag store after the 7th read are shown below. The only part shown in a tag store entry is the tag. You can assume it is valid. Many of the tag store entries unnecessary to solving this problem are not shown in order to reduce the clutter.

BEFORE

	Way 0	Way 1	LRU
Set 0	1	4	1
Set 1			0
Set 2	6	5	0
Set 3			0
Set 4			0
Set 5			0
Set 6			0
Set 7			0
Set 8			0
Set 9			0
Set 10	9	8	0
Set 11			0
Set 12			0
Set 13			0
Set 14			0
Set 15			0

AFTER

	Way 0	Way 1	LRU
Set 0	1	4	1
Set 1			0
Set 2	1	4	1
Set 3			0
Set 4			0
Set 5			0
Set 6	3	4	0
Set 7			0
Set 8			0
Set 9			0
Set 10	9	8	0
Set 11			0
Set 12			0
Set 13			0
Set 14			0
Set 15			0