The University of Texas at Austin
EE460N/EE382N.1 Exam 2 (Fall 2017)

Mattan Erez, Sarbartha Banerjee, Madhuri Gontala, Gayatri Sivaraman

Your signature is your promise that you have **not cheated** and will not cheat on this exam, nor will you help others to cheat on this exam. Cheating means violating the explicit bullets below, or their spirit.

Signature: | Name: | EID:

Note: Please be sure that your answers to all questions (and all supporting work that is required) **are contained in the box provided for each question;** we reserve the right to ignore anything else.

Note: **"I DON'T KNOW"** is a valid answer that automatically gives you 15% of the maximum possible grade. Be sure to either erase, or cross-out everything else and write I DON'T KNOW in, large, capital letters. A blank answer does not get you the 15%. No partial I DON'T KNOW credit (only on entire questions/parts that have a marked point value on the exam)

Note: Please put your name at the top of each page of the exam!

Note: For all questions, unless otherwise stated, find the most efficient solution. Efficiency counts.

Note: The exam is open book, open notes, bring anything type of exam, but there are constraints:

- **Do not ask anyone or anything for help. You may only search for text in PDFs or other material that you have downloaded.**
- **As a specific example of the above, no web searches.**
- **Do not use a calculator, phone, tablet, laptop, watch, … to compute anything. Not simple arithmetic and not complex simulations.**
- **Do not share any material related to the exam with anyone (or upload it anywhere). The exam will be made public soon.**
- **You should have no real need to type anything (or talk to a device), hence, significant typing is not allowed (minor Ctrl+F searches are OK).**

The exam has a total of XXXX questions and 7 pages. It's a good idea to read through first some questions might be easier for you than others.

For each page that doesn't include your EID you may be deducted 1 point!

Question 1 (20 pt):

Consider the code below which executes on the 8-stage, in-order, single functional unit pipeline shown below. Cache access requires 10ns, execution unit latency totals 20ns, and main memory latency is 100ns. The cache is 16KiB, fully associative, 2B cachelines, and is blocking.

Rank the following pipeline improvements in order of most to least beneficial: add 2 delay slots to all branches, add a bypass that forwards values from the end of functional unit execution to the inputs of the functional, and a main memory prefetcher that can keep memory busy. Explain your answers.

F	F	D	E	E	M	M	W

```
  R1 = ArrayA
  R3 = ArrayB
  R0 = 0xffff
loop
  R2 = LD [R1]
  MEM[R3] = R2
  R1 = R1 + 2
  R3 = R3 + 2
  R0 = R0 - 1
  Bnp loop ; was Bp at exam time by mistake
```

Rank	Mechanism	Explanation
1	**Prefetch**	**No spatial locality in cache because of short lines and no temporal locality because of large arrays. So every access is a cache miss. A good prefetcher will get the data into the cache and eliminate all the cache misses.**
2	**Delay slots**	**Will save 2 bubbles per loop iteration.**
3	**Bypass**	**Doesn't help at all as stated. Will help a little if assuming can forward from memory (not stated) or if assuming needing to forward for branch resolution (ambiguous in the question).**

Question 2 (40 pt):

An LC3B system has a 2-level VAX type virtual memory system. The whole address space is partitioned as follows:

System Space: **0x0000 – 0x7FFF** User Space: **0x8000- 0xFFFF**

Initially only the System page table and the trap vector table is loaded into the physical memory. The page size is 128 bytes and the PTE is 2 bytes. The physical memory has 32 frames. The user region page table base register address is 0x3000.

The questions below relate to the following code, which is run on the system:

```
     .ORIG  x81F6
     LEA R0, LOC
     LDW R1,R0,#0
      BR LBL
LOC  .FILL  xFA24
LBL  LDW R3,R1,#0        ; R3 is xFA42 after this instruction
     STW R2,R3,#0
     HALT
     .END
```

Part a (5 pt) :

How many frames are needed to load the system page table if it needs to hold translations for the **entire** system space? Assume that the page table is page aligned (i.e., starting address is multiple of the page size).

4

Part b (18 pts):
List the number of page **faults** for each **instruction** in the code in the table below:

Instruction	Number of Page faults
`LEA R0, LOC`	2(Process page table + instruction address)
`LDW R1,R0,#0`	0
`BR LBL`	0
`LDW R3,R1,#0`	2(Process page table + data)
`STW R2,R3,#0`	1(Next page)
`HALT`	0

Part c (9 pt):
What are the virtual pages referenced by the program? (Start address of the page will also suffice as an answer)

System Pages	User Pages
X3000 X3180 x0000	X8180 xFA00 x8200

Part d: (8 pts):
If you are given a freedom to change the placement of the code in user space memory (. ORIG) , can you reduce the number of page faults? Explain.

If we place everything(instruction + data) in a single page such that there PTEs reside in the same page.

Question 3 (40 pt):

Consider a system with an unpipelined processor, a main memory, and a network interface. All three components share a bus. Details of the components below. If little/big endian matters, **choose little-endian.**

The processor:

- Has 8 16-bit registers.
- **Executes instructions at a fixed rate of 1μs (10^{-6} seconds) per instruction, except for load/store.**
- A load/store that hits the cache finishes like all other instructions; but **a cache miss requires additional time -- that time is precisely the time of reading the cacheline over the bus**.
- No virtual memory.
- Has a direct-mapped PIPT cache with a total of 4 4B-long cachelines (the cache uses physical addresses only). The cache is read and write allocate.
- The cache controller snoops the bus and implements the MSI coherence protocol.

Main memory:

- Main memory capacity is 64KiB, it serves request in 2B granularity (2B per memory request), and each request has a memory read/write latency of 1μs.
- Memory is initialized to all-zero except for the program text (binary).

The network interface card (NIC):

- Sends and receives messages that are 4B long.
- Has a message buffer memory-mapped to physical address 0x4000.
- Has a send flag that is memory mapped to address 0x5008; if the processor writes any non-zero value to PA 0x5008, the NIC initiates a read from the message buffer in memory and sends it when the read is complete.
- In this system, a message will be received at the NIC exactly 10μs after the NIC sends a message. The message will reflect the previous message sent (interpreted as an unsigned int) multiplied by 16. The NIC will write that message to the message buffer in memory immediately when received. **Sending and receiving over the network is not a bus transaction.**

The bus:

- The bus and cache controller implement the snooping MSI protocol.
- **Physical addresses in the range 0x5000 - 0x6000 are never cached.**
- The data bus operates at 1MHz (10^6 cycles per second).
- The bus is 8B wide for data and 3B wide for addresses/command.
- Bus transactions are pending.
- There are dedicated control lines between each device (processor, memory, and NIC) to the centralized bus arbiter. The arbiter prioritizes the NIC over the processor. Delayed requests are granted automatically as soon as possible.

Part a (5 pt):

What is the minimal amount of time the processor requires to execute a load that misses the cache?

1us - Sending command to memory
1us - memory latency
1us - 2B of data sent back
Transaction = 4B
Total time = (1us+1us+1us)*2 = 6us

If cache miss time is considered, total time = 1us(miss) + 6us = 7us

Part b (8 pt):
What is the maximum amount of time the processor requires to execute a load that misses the cache?

Graded Solution:
NIC initiates read same time as processor requests for memory, NIC is serviced first
NIC Read: (1us+1us+1us)*2 = 6us
Total time = NIC read + PartA = 12us

Better Solution:
Cache capacity miss leading to Writeback of cache(6us) + NIC Read/Write time(6us) + PartA = 18us

Partial credit:
NIC initiates read same time as processor requests for memory, NIC is serviced first
NIC Read: (1us+1us+1us)*2 = 6us
NIC receives data = 10us
NIC writes back to x5008 = 6us
Total time = 6us + 10us + 6us+ Part a = 28us

Part c (12 pt): Show the cache state **after the entire** command sequence executes. The cache is initially empty with all lines invalid and storing all zeros. If a cacheline is invalid, write down the value it had last.

Instruction

`R0 = 0x1234`
`R1 = LDW[0x1000]`
`R2 = LDW[0x600E]`
`STW [0x4000]<-R0`
`STW [0x4002]<-R1`
`STW [0x5008]<-R0`
`R2 = LDW[0x600C]`

4 cachelines:

MSI	Tag	Value
S	400	0x00001234
I		0x00000000
I		0x00000000
S	600	0x00000000

Part d (15 pt): Show the cache state **after the entire** command sequence executes. The cache is initially empty with all lines invalid and storing all zeros. If a cacheline is invalid, write down the value it had last.

Instruction

`R0 = 0x1234`
`R1 = LDW[0x1000]`
`R2 = LDW[0x101A]`
`STW [0x4000]<-R0`
`STW [0x4002]<-R1`
`STW [0x5008]<- 1`
`R2 = LDW[0x1018]`
`R3 = LDW[0x102E]`
`R4 = LDW[0x1044]`
`R5 = LDW[0x1066]`

4 cachelines:

MSI	Tag	Value
I→ M→ S→ I		0x00001234
S	104 → 106	0x00000000
S	101	0x00000000
S	102	0x00000000