Introduction to Computer Organization – Fall 2024

Homework 4

Due: @11:55PM, Monday, December 9th

Name: _____Qiulin Fan______ Uniqname: _____rynnefan_______

1. Submit a pdf of your typed or handwritten homework on Gradescope.

2. Your answers should be neat, clearly marked, and concise. Typed work is recommended, but not required unless otherwise stated. Show all your work where requested, and state any special or non-obvious assumptions you make.

3. You may discuss your solution methods with other students, but the solutions you submit must be your own.

4. **Late Homework Policy:** Submissions turned in by 1:00 am the next day will be accepted but with a 5% penalty. Assignments turned in between 1:00 am and 11:55 pm will get a 30% penalty, and any submissions made after this time will not be accepted.

5. When submitting your answers to Gradescope you need to indicate what page(s) each problem is on to receive credit. The grader may choose not to grade the homework if answer locations are not indicated.

6. After each question (or in some cases question part), we've indicated which lecture number we expect to cover the relevant material. So "**(L7)**" indicates that we expect to cover the material in lecture 7 (but depending on your lecture, it may have been covered later)

7. **The last question is a group question**.

○ You may do it in a homework group of up to two students including yourself (yes, you can do them by yourself if you wish).

○ If you work in a group of two for these questions, list the name of the student you worked with in your assignment. Further, we suggest that you not split these problems up but rather work on the problem as a group.

○ Turn these group questions in as part of your individual submission.

○ **For these questions (and these questions ONLY) you are allowed to copy/paste solutions from the other student in your homework group.**

○ It is an honor code violation if a student is listed as contributing who did not actually participate in working on that problem.

1111 1110
nor -5 = 1111 1011
→ 0

0000 0100
→ 1111 1011
→ 1111 1100 = -4
nor 1111 1110
= 1

Problem 1: Out of Control [16 points]

For the questions below, reference the following piece of LC2K assembly code: **(L15)**

```
        lw      0       1       pos1
        lw      0       2       neg1
        lw      0       3       count
        nor     1       1       4
loop    beq     0       3       fin             //beq0
        nor     3       3       5
        nor     4       5       5
        beq     0       5       even            //beq1
        add     6       1       6
        beq     0       0       next            //beq2
even    add     7       1       7
next    add     3       2       3
        beq     0       0       loop            //beq3
fin     halt
count   .fill 5
pos1    .fill 1
neg1    .fill -1
```

1: 1
2: -1
3: 5
4 = 1 nor 1 = -2

N	N	N	N	N	T
5: -6	5: -5	5: -4	5: -3	5: -2	
5: 1	5: 0	5: 1	5: 0	5: 1	
N	T	N	T	N	
6: 1		6: 2		6: 1	
T		T		T	
	7: 1		7: 2		
3: 4	3: 3	3: 2	3: 1	3: 0	
T	T	T	T	T	

0000 0010
↓
1111 1101 + 1
-2 = 1111 1110
nor -6 = 1111 1010
0000 0001 = 1

1. Write the sequence of branch decisions for each beq instruction. Let "taken" be denoted as "T" and "not taken" as "N". For example, a beq that is taken twice and then not taken once would have a sequence TTN. **[4]**

beq0	NNNNNT
beq1	NTNTN
beq2	TTT
beq3	TTTTT

8N 11T

How many extra cycles are added due to the following schemes by squashes, assuming the 5-stage pipeline from lecture? Show your work.

2. Speculate and Squash: Predict always taken **[2]**

8N 11T

⇒ +8×3 = 24 cycles

3. Speculate and Squash: Predict backwards taken, forwards not taken **[3]**

beq 0 : forword, 5N 1T, +3 cycles
beq 1 : forward, 3N 2T, +6 cycles
beq 2 : forward, 3T, +9 cycles
beq 3 : backwards 5T ✓

+18 cycles in total

4. Dynamic branch predictor with separate 1-bit counter for each branch: Initialized to taken **[3]**

beq 0 : NNNNNT
TNNNNN predicted, +2×3=6 cycles

beq 1 : NTNTN
TNTNT predicted, +5×3=15 cycles

beq 2,3: ✓

total: +21 cycles

5. Dynamic branch predictor with separate 2-bit counter for each branch: Initialized to strongly taken **[3]**

beq 0 : NNNNNT
TTNNNN predicted, +3×3 = 9 cycles

beq 1 : NTNTN
TTTTT predicted, +3×3 = 9 cycles

beq 2,3 : ✓

total : +18 cycles

Problem 2: A real page turner [20 points] (L22)

Consider a 42-bit, byte-addressable system that uses virtual memory. The system has a maximum of 128 GB of physical memory installed with a page size of 8 KB.

1. How many bits are used for the page offset? How many bits are used to index into the page table(s)? **[4]**

   $2^3 \times 2^{10}$

   Page Offset: 13 Virtual Page Number Bits: 29

2. How many **virtual** pages exist in the system? **[4]**

   2^{29}

   $2^7 \times 2^{30} / 2^{13} = 2^{24}$

3. How many **physical** pages exist in the system? **[4]**

   2^{24}

4. If the system has a single-level page table with 8B entries, how large must the page table be to map all the system's virtual memory? **[4]**

   $2^{29} \times 8B = 2^{32}B$ (= 4 GB)

5. If we switch to a hierarchical page table such that each page table level must fit in a single page, how many page table levels would we need to map all of the system's virtual memory? Assume page table entries are still 8B each. **[4]**

   each page: 8KB ⇒ has $8 \times 2^{10} / 8 = 2^{10}$ entries

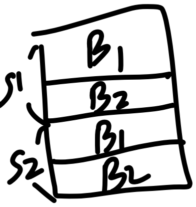
   So the i^{th} level pages can fill a total of $2^{10(i-1)} \times 8KB$

   $= 2^{10i+3}$ B VM

   $2^{10\times2+3} < 2^{29}$, $2^{10\times3+3} > 2^{29}$

   So we need 3 levels

Problem 3: Miss Me With That [20 points] (L20)



Consider a byte-addressable system which includes a cache with the following specifications:

- 2-Way Set-Associativity
- Block Size: 8 bytes $2^3 B$
- Cache Size: 32 bytes
- LRU replacement policy

4	1	3
tag	set	offset

Given the following sequence of cache accesses, determine whether each access is a hit or a miss and then classify each of the misses as one of compulsory, capacity, or conflict. Additionally, break down each address to show the tag and set index in order to help see what is in the cache. The last three columns are optional.

Address 0011|1|011	Tag (Binary)	Set Index	Hit / Miss?	Type (N/A if Hit)	Infinite Cache	Fully Associative	Set Associative
0x3B 0100|0|000	*0011* *|1|011*	*1*	*Miss*	*Compulsory*	*x*	00111 *x*	*x*
0x40	0100	0	Miss	Compulsory	X	00111 01000 X	X
0x03 0000|0|011	0000	0	Miss	Compulsory	X	00111 01000 00000 X	X
0x5B 0101|1|011	0101	1	Miss	Compulsory	X	X	X
0x00	0000	0	Hit	N/A	✓	00111 01000 00000 01011 ✓	✓
0x2F 0010|1|111	0010	1	Miss	Compulsory	X	00101 X	X
0x3E 0011|1|110	0011	1	Miss	Capacity	✓	00101 00111 X	X
0x59 0101|1|001	0101	1	Miss	Conflict	✓	00000 ✓	X
0x69 0110|1|001	0110	1	Miss	Compulsory	X	01011 00101 00111 01101 01011 X	X
0x4E 0100|1|110	0100	1	Miss	Compulsory	X	01001 X	X
0x5D 0101|1|101	0101	1	Miss	Conflict	✓	01001 00111 01101 01011 ✓	X
0x46 0100|0|110	0100	0	Hit	N/A	✓	01001 01000 01101 01011 X	✓
0x24 0010|0|100	0010	0	Miss	Compulsory	X	00100 X	X
0x07 0000|0|111	0000	0	Miss	Capacity	✓	X	X

Problem 4: Two-level page table [20 points] (L22)

You are checking the correctness of a system with virtual memory and no caches. The system uses a 2-level page table scheme with the following partitioning for 12-bit virtual addresses:

L1 Page Table Index	L2 Page Table Index	Page Offset
Bits 11-8	Bits 7-4	Bits 3 - 0

4 $2^4 = 16$ entries

The page table translates virtual addresses into physical addresses of the following form:

Physical Page Number	Page Offset
Bits 10 - 4 2^7	Bits 3 - 0

Each page table entry is **1 byte** wide, and it is stored in (physical) memory. The format of each page table entry is as follows:

Valid bit	Page Number
Bit 7	Bits 6 - 0

1000

2

(Note: this implies that level-2 page tables must start at a page boundary, since we are only storing a page number in level-1 entries, rather than a full address). A snippet of the machine's physical memory is given in the next page (contents are laid out 16 bytes per row, the first entry being the lowest-order byte. For example, address 0x1 contains the byte 0xf2). You also know that the page table **base register** points to physical address **0x110**.

Translate the two virtual addresses in the next page into physical addresses. Specify the address (not the index!) of the Level 1 (L1) page table entry of interest and its value, and do the same for second-level page tables. Write "–" to indicate an invalid or unknown entry/address. [Hint: You need to use page table entry size and page table index to determine the location of a page table entry in memory.]

Virtual Address	Address of L1 Page Table Entry +3	Content of L1 Page Table Entry 6 1|0000110	Address of L2 Page +2 Table Entry 6th L2 table	Content of L2 Page Table Entry	Physical Address	Content of Physical Address
0x32f	0x113	86	0x62	82	2f	99
0x1c6	0x111	00	–	–	–	–

0 –

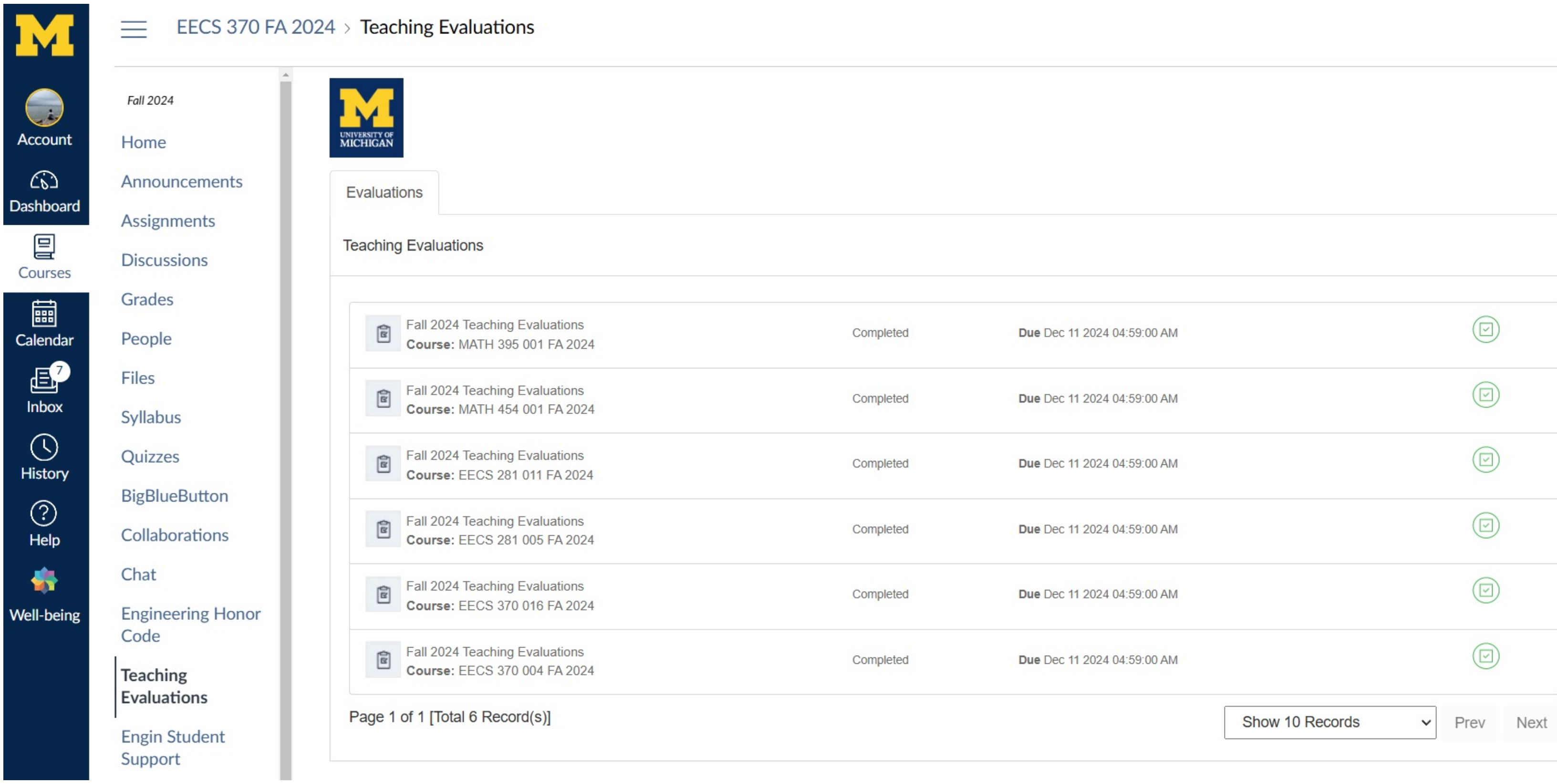
Snippet of physical memory (All values in hexadecimal):

(page table)

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+a	+b	+c	+d	+e	+f
0x0	f0	f2	00	00	00	00	00	00	00	a4	40	23	44	12	12	12
0x10	86	8c	c9	4f	00	3f	42	00	00	00	8a	a3	9f	91	00	19
0x20	00	af	00	14	00	12	00	00	5f	00	40	00	00	90	33	99
0x30	c6	e8	39	a5	3a	20	b0	ff	72	79	e7	4e	be	41	70	c4
0x40	6a	91	f2	40	94	62	19	9c	97	04	9f	95	cd	30	1c	9e
0x50	00	86	19	4a	50	b0	3f	16	e8	f9	4e	06	1a	75	da	e2
0x60	80	99	82	00	8f	90	91	92	c4	d3	00	00	34	00	00	00
0x70	d2	ff	78	ae	e8	cb	57	56	f8	b3	55	08	d3	bd	9c	cd
0x80	46	30	c9	4f	44	3f	42	70	22	62	8a	b9	78	28	40	19
0x90	61	07	11	3	00	00	00	ca	00	00	00	00	00	80	09	00
0xa0	00	b8	00	ba	00	b9	00	00	00	40	00	0b	c0	00	aa	00
0xb0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0xc0	00	00	c3	b4	b5	00	b6	00	00	00	00	00	00	00	00	00
0xd0	54	94	09	68	0d	fd	db	46	6e	85	dd	a3	01	f7	f7	13
0xe0	ad	ac	10	a1	ed	83	92	42	99	40	c5	ca	df	68	20	8f
0xf0	12	d1	b3	44	00	55	00	00	00	b4	9c	ae	a0	97	b0	c2
0x100	1f	00	02	00	e3	00	00	54	00	00	20	34	76	00	50	30
0x110	90	00	00	86	00	00	a3	a2	a1	c0	00	00	00	00	00	00
0x120	6f	ef	01	aa	70	22	b7	30	ce	67	05	97	b0	b7	5d	6d
0x130	00	34	00	00	55	55	55	67	00	88	24	00	4a	01	00	3c

Problem 5: Course Surveys [5 points]

1. Take a screenshot that shows you did at least one end-of-term course evaluation for EECS 370. We really do read all of these and that can be very useful in helping with future terms. **[5]**



Teaching Evaluations

Fall 2024 Teaching Evaluations **Course:** MATH 395 001 FA 2024	Completed	**Due** Dec 11 2024 04:59:00 AM	✓
Fall 2024 Teaching Evaluations **Course:** MATH 454 001 FA 2024	Completed	**Due** Dec 11 2024 04:59:00 AM	✓
Fall 2024 Teaching Evaluations **Course:** EECS 281 011 FA 2024	Completed	**Due** Dec 11 2024 04:59:00 AM	✓
Fall 2024 Teaching Evaluations **Course:** EECS 281 005 FA 2024	Completed	**Due** Dec 11 2024 04:59:00 AM	✓
Fall 2024 Teaching Evaluations **Course:** EECS 370 016 FA 2024	Completed	**Due** Dec 11 2024 04:59:00 AM	✓
Fall 2024 Teaching Evaluations **Course:** EECS 370 004 FA 2024	Completed	**Due** Dec 11 2024 04:59:00 AM	✓

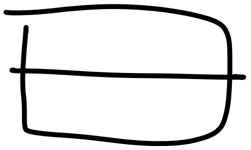
Page 1 of 1 [Total 6 Record(s)]

Problem 6: I'm having cache flow problems [20 points, Group] (L19)

Consider the following access pattern: A, B, C, D, A. Assume that A, B, C and D are memory addresses each of which are in a different block of memory. Further, assume A, B, C and D are generated in a uniformly random way (each block is equally likely) and that a true LRU replacement algorithm is used. *To receive credit you must briefly show your work*.

What is the probability that the second instance of "A" will be a hit if:

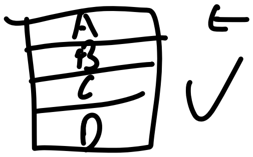
1. The cache has 2 lines and is fully-associative **[3]**.



All in one set. The C.D will occupy the cach anyway so impossible that A is not evicted.

So Possibility of hit is 0

2. The cache has 4 lines and is fully-associative **[3]**.



A

B

C

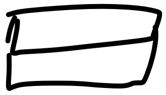
D

← hit

possibility of hit is 1

3. The cache has 2 lines and is direct-mapped **[3]**.

B, C, D has 50% possibility to be in the same set as A



So $P(A\ \text{hit}) = P(B, C, D \text{ all in the other set})$

$= 0.5^3 = 0.125$

4. The cache has 4 lines and is direct-mapped, *you may leave your answer in a fraction.* **[5]**

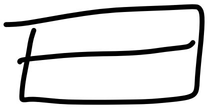
$P(A\ \text{hit}) = P(B, C, D \text{ not in same set as } A)$

$= 0.75^3 = \frac{3^3}{4^3} = \frac{27}{64}$

Now, assume the access pattern is this instead: “A, B, C, C, A”. Again, Assume that A, B, and C are memory addresses each of which are in a different block of memory. Further, assume A, B and C are generated in a uniformly random way (each block is equally likely) and that a true LRU replacement algorithm is used.
What is the probability that the second instance of "A" will be a hit if:

5. The cache has 2 lines and is direct-mapped **[3]**.

$P(A \text{ hit})$

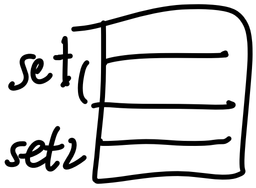


$= P(B, C \text{ not in the same set as } A)$

$= \underline{0.25}$

6. The cache has 4 lines and is two-way set associative? **[3]**

$P(A \text{ hit})$



$= P\left(\begin{array}{c}\text{at least one of } B, C \\ \text{not in same set} \\ \text{as } A\end{array}\right)$

$= 1 - P(B, C \text{ both in same set as } A)$

$= 1 - 0.5 \times 0.5 = \underline{0.75}$