L26_1 Final_Exam-Review

EECS 370 – Introduction to Computer Organization – Fall 2020

Learning Objectives

- To understand exam logistics
  - Logistics: the detailed coordination of a complex operation involving many people, facilities, or supplies.

- To review concepts and their applications to concrete problems.

- To see the process of solving exam questions using problems representative of exam questions.

- The worksheets for this lecture are the past exams.
- **There is no participation quiz for L26**

Preparing for the Exam

- The Winter 2020 exam is available on Gradescope as a practice exam
  - https://www.gradescope.com/

- Take the exam!

  - This is the best practice for exam questions and the format of the exam

- See the note on exam logistics on Google drive
  - https://drive.google.com/drive/folders/1N-p4sFqkQCtkl6aNkctTLNeTQbJlg1Sx

Announcements

- Final Exam is on Monday, December 14th from 10:30 am to 12:30 pm
  - If you have requested an alternate you will receive confirmation from 370 staff

- 2 hours to finish (unless you contacted us about accommodations)

- Open note / compiler /internet (but no soliciting help)

Preparation Strategy

- Use your discussion pods as a study group!
  - Grade each other's exams
  - Write questions for each other
  - Talk over stuff you do not understand
- Come to review sessions!
- Know how to solve the homework problems
- Take past exams
  - **Complete without looking at the solutions!**
- Take the practice exam on Gradescope

Exam Strategy

- Pace yourself
- Remember to show your work for partial credit

- **Do not spend all your time on a single question**
  - If you don't understand a question, move ahead & come back to it later

- Answer as many questions as possible
  - Give yourself a few minutes at the end to check your work and/or attempt to solve easy questions that you have not attacked yet
- Show work where possible

Important Topics

- Syllabus: All lectures

- Covers **all** course material with **emphasis on Lectures 12-24**
  - Pipelining, data, and control hazards
  - Caches
  - Virtual memory
  - Performance

- Material presented in lectures, homeworks, discussions, projects

Review Questions

If a topic is not covered in this review, it does not imply that it is not important!

Logistics

- There are 3 videos for lecture 26
  - L26_1 – Final_Exam-Review
  - L26_2 – Final_Exam-Review-Questions_1C+VM
  - L26_3 – Final_Exam-Review-Questions_BP+MLPT

- There is no worksheet for lecture 14
  1. The worksheets are the past exams!
  2. https://www.eecs.umich.edu/courses/eecs370/eecs370.f20/exams/

L26_2 Final_Exam-Review-Questions_MC+VM

EECS 370 – Introduction to Computer Organization – Fall 2020

Review Lecture Strategy

- Start with the slides, available on Google drive
- Work one problem at a time, one part at a time
  - Solve the problem first
  - Compare your answers to solutions on slides
  - If you get all of them correct and feel confident, skip the part of the recording
    - Invest the time gained in taking a previous semester exam
  - If you did not answer correctly and/or are not feeling confident about the question, watch that part of the recording

Problem 1: Multiple Choice

1. In ARM, the first 4 parameters to a function are stored in registers and the remaining parameters are stored in the ***stack / heap / data / virtual*** section of memory.

2. ***Moore's / Mealy's / Dennard's / Immerman's*** Law states that the number of transistors on a single chip will double every 2 years.

3. All instructions in a ***CISC / RISC / multi-core*** architecture (usually) have the same length.

Problem 1: Multiple Choice

4. The ***compiler / assembler / linker / loader*** is used to combine multiple object files into an executable program.

5. The ***compiler / assembler / linker / loader*** is used every time an executable program is run.

Problem 1: Multiple Choice

6. The output of ***combinational / sequential*** circuits depends exclusively on the current input.

7. Functions that don't call any functions (leaf functions) need not save and restore ***caller save / callee save / odd numbered*** registers.

8. If a cache includes dirty bits, then it must be using a ***write-back / write-through / allocate-on-write / not allocate-on-write*** policy.

Problem 1: Multiple Choice

9. The ***single-cycle / multi-cycle / pipelined*** datapath has the lowest CPI among the three types of datapath discussed in class.

10. A 1-way set associative cache is the same as a ***fully associative / direct mapped / virtually addressed / physically addressed*** cache.

11. Direct-mapped caches reduce the cost of finding the correct data by minimizing the number of **block offset / line index / tag / LRU** comparisons.

Problem 2 – Virtual Memory

Assume the following:

Virtual address size : 128 bytes; **Page size** : 16 bytes

Physical memory size : 8 pages; **One-level page table of size** : 16 bytes

Page replacement policy : LRU

Notes:

- On a page fault, the page table is updated before allocating a physical page.
- If more than one free page is available, the smallest physical page number is chosen.
- Physical page #0 is reserved for the operating system (OS). It cannot be replaced.
- If no mapping is found for a given virtual page assume the data is brought in from the disk.

Problem 2 – Virtual Memory (cont.)

a. The initial state of physical memory is shown on the left. Complete the page table for Process ID 11 (PID: 11) on the right.

Physical Page # (PPN)	Memory Contents
0x0	Reserved for OS
0x1	Page Table of PID 11
0x2	PID 11: VPN 0
0x3	
0x4	PID 11: VPN 4
0x5	PID 11: VPN 7
0x6	
0x7	

Page Table of PID:11

Virtual Page # (VPN)	Valid	Physical Page # (PPN)
0x0		
0x1		
0x2		
0x3		
0x4		
0x5		
0x6		
0x7		

Problem 2 – Virtual Memory (solution)

a. The initial state of physical memory is shown on the left. Complete the page table for Process ID 11 (PID: 11) on the right.

Physical Page # (PPN)	Memory Contents
0x0	Reserved for OS
0x1	Page Table of PID 11
0x2	PID 11: VPN 0
0x3	
0x4	PID 11: VPN 4
0x5	PID 11: VPN 7
0x6	
0x7	

Page Table of PID:11

Virtual Page # (VPN)	Valid	Physical Page # (PPN)
0x0	1	0x2
0x1	0	
0x2	0	
0x3	0	
0x4	1	0x4
0x5	0	
0x6	0	
0x7	1	0x5

Problem 2 – Virtual Memory (cont.)

b. Complete the following table for the given sequence of virtual address requests. Assume that the initial physical memory and page table state is shown in part a, above. *Process 7 begins at Time 1. Process 11 begins before Time 0, and ends after Time 5.*

Physical Page # (PPN)	Memory Contents
0x0	Reserved for OS
0x1	Page Table of PID 11
0x2	PID 11: VPN 0
0x3	Page Table of PID 7
0x4	PID 11: VPN 4
0x5	PID 11: VPN 7
0x6	
0x7	

Time	PID	Virt. Addr (VA)	Virtual Page # (VPN)	Physical Page # (PPN)	Page Fault? (Y/N)	Physical Address (PA)
0	11	0x0A				
1*	7	0x0A				
2	7	0x0B				
3	11	0x21				
4	7	0x74				
5*	7	0x35				
6	7	0x20				

Problem 2 – Virtual Memory (cont.)

b. Complete the following table for the given sequence of virtual address requests. Assume that the initial physical memory and page table state is shown in part a, above. **Process 7 begins at Time 1. Process 11 begins before Time 0, and ends after Time 5.*

Physical Page # (PPN)	Memory Contents
0x0	Reserved for OS
0x1	
0x2	
0x3	
0x4	
0x5	
0x6	
0x7	

Time	PID	Virt. Addr (VA)	Virtual Page # (VPN)	Physical Page # (PPN)	Page Fault? (Y/N)	Physical Address (PA)
0	11	0x0A				
1*	7	0x0A				
2	7	0x0B				
3	11	0x21				
4	7	0x74				
5*	7	0x35				
6	7	0x20				

Problem 2 – Virtual Memory (cont.)

b. Complete the following table for the given sequence of virtual address requests. Assume that the initial physical memory and page table state is shown in part a, above. *Process 7 begins at Time 1. Process 11 begins before Time 0, and ends after Time 5.*

Physical Page # (PPN)	Memory Contents
0x0	Reserved for OS
0x1	Page Table of PID 11
0x2	PID 11: VPN 0
0x3	Page Table of PID 7
0x4	PID 11: VPN 4
0x5	PID 11: VPN 7
0x6	
0x7	

Time	PID	Virt. Addr (VA)	Virtual Page # (VPN)	Physical Page # (PPN)	Page Fault? (Y/N)	Physical Address (PA)
0	11	0x0A	0x0	0x2	N	0x2A
1*	7	0x0A	0x0	0x6	Y	0x6A
2	7	0x0B	0x0	0x6	N	0x6B
3	11	0x21	0x2	0x7	Y	0x71
4	7	0x74	0x7	0x4	Y	0x44
5*	7	0x35	0x3	0x5	Y	0x55
6	7	0x20	0x2	0x1	Y	0x10

For More Depth…

- Lecture 22 – example problem at slide #28

- Test your knowledge:
  - Winter 2019, Problem 12

Logistics

- There are 3 videos for lecture 26
  - L26_1 – Final_Exam-Review
  - L26_2 – Final_Exam-Review-Questions_1C+VM
  - L26_3 – Final_Exam-Review-Questions_BP+MLPT

- There is no worksheet for lecture 14

1. The worksheets are the past exams!
2. https://www.eecs.umich.edu/courses/eecs370/eecs370.f20/exams/

L26_2 Final_Exam-Review-Questions_BP+MLPT

EECS 370 – Introduction to Computer Organization – Fall 2020

Problem 3: Branch Prediction

Consider the following C code and corresponding LC2K assembly. Assume it is executed on a pipelined data-path with branch speculation discussed in class.

a. How many branch instructions are executed for a single run of the program?

```
int i, j, x = 0;
for (i=0; i != 30; i++) {
    for (j=0; j != 2; j++) {
        x++;
    }
}
```

```
        lw   0  1  cnt1
        lw   0  2  cnt2
        lw   0  3  one
outer   noop
        beq  0  1  exit        //B1
        noop
        lw   0  2  cnt2
inner   beq  0  2  done        //B2
        add  4  3  4
        add  2  3  2
        beq  0  0  inner       //B3
done    add  1  3  1
        beq  0  0  outer       //B4
exit    halt
cnt1    .fill 30
cnt2    .fill 2
one     .fill -1
```

Problem 3: Branch Prediction

Consider the following C code and corresponding LC2K assembly. Assume it is executed on a pipelined data-path with branch speculation discussed in class.

a. How many branch instructions are executed for a single run of the program?

```
int i, j, x = 0;
for (i=0; i != 30; i++) {
    for (j=0; j != 2; j++) {
        x++;
    }
}
```

```
        lw   0  1  cnt1
        lw   0  2  cnt2
        lw   0  3  one
outer   noop
        beq  0  1  exit        //B1
        noop
        lw   0  2  cnt2
inner   beq  0  2  done        //B2
        add  4  3  4
        add  2  3  2
        beq  0  0  inner       //B3
done    add  1  3  1
        beq  0  0  outer       //B4
exit    halt
cnt1    .fill 30
cnt2    .fill 2
one     .fill -1
```


B1: 31 B2: 90 B3: 60 B4: 30

Problem 3: Branch Prediction

Consider the following C code and corresponding LC2K assembly. Assume it is executed on a pipelined data-path with branch speculation discussed in class.

b. How many branches are predicted correctly if we predict Always-Not-Taken?

```
int i, j, x = 0;
for (i=0; i != 30; i++) {
    for (j=0; j != 2; j++) {
        x++;
    }
}
```

```
        lw   0  1  cnt1
        lw   0  2  cnt2
        lw   0  3  one
outer   noop
        beq  0  1  exit        //B1
        noop
        lw   0  2  cnt2
inner   beq  0  2  done        //B2
        add  4  3  4
        add  2  3  2
        beq  0  0  inner       //B3
done    add  1  3  1
        beq  0  0  outer       //B4
exit    halt
cnt1    .fill 30
cnt2    .fill 2
one     .fill -1
```

Problem 3: Branch Prediction

Consider the following C code and corresponding LC2K assembly. Assume it is executed on a pipelined data-path with branch speculation discussed in class.

b. How many branches are predicted correctly if we predict Always-Not-Taken?

```
int i, j, x = 0;
for (i=0; i != 30; i++) {
    for (j=0; j != 2; j++) {
        x++;
    }
}
```

```
        lw   0  1  cnt1
        lw   0  2  cnt2
        lw   0  3  one
outer   noop
        beq  0  1  exit        //B1
        noop
        lw   0  2  cnt2
inner   beq  0  2  done        //B2
        add  4  3  4
        add  2  3  2
        beq  0  0  inner       //B3
done    add  1  3  1
        beq  0  0  outer       //B4
exit    halt
cnt1    .fill 30
cnt2    .fill 2
one     .fill -1
```


Total executions: B1: 31 B2: 90 B3: 60 B4: 30

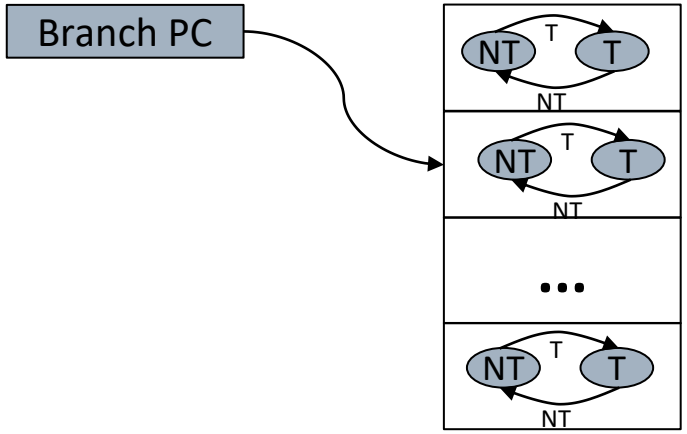
NT executions: B1: 30 B2: 60 B3: 0 B4: 0

Problem 3: Branch Prediction

c. How many branches are predicted correctly if we use local last time 1-bit predictor?

Local 1-bit Predictor: In this prediction scheme, the PC of a branch instruction is used to index into a table. Each table entry for a branch is a 1-bit predictor, which predicts the same outcome as last time for that branch. Assume that the entries are initialized to Not Taken.

Local Last Time Predictor Table




```
int i, j, x = 0;
for (i=0; i != 30; i++) {
    for (j=0; j != 2; j++) {
        x++;
    }
}
```

```
        lw   0  1  cnt1
        lw   0  2  cnt2
        lw   0  3  one
outer   noop
        beq  0  1  exit        //B1
        noop
        lw   0  2  cnt2
inner   beq  0  2  done        //B2
        add  4  3  4
        add  2  3  2
        beq  0  0  inner       //B3
done    add  1  3  1
        beq  0  0  outer       //B4
exit    halt
cnt1    .fill 30
cnt2    .fill 2
one     .fill -1
```

Problem 3: Branch Prediction

c. How many branches are predicted correctly if we use local last time 1-bit predictor?

Local 1-bit Predictor: In this prediction scheme, the PC of a branch instruction is used to index into a table. Each table entry for a branch is a 1-bit predictor, which predicts the same outcome as last time for that branch. Assume that the entries are initialized to Not Taken.

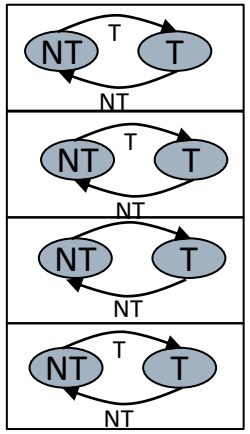
Local Last Time Predictor Table

//B1

//B2

//B3

//B4




```
int i, j, x = 0;
for (i=0; i != 30; i++) {
    for (j=0; j != 2; j++) {
        x++;
    }
}
```

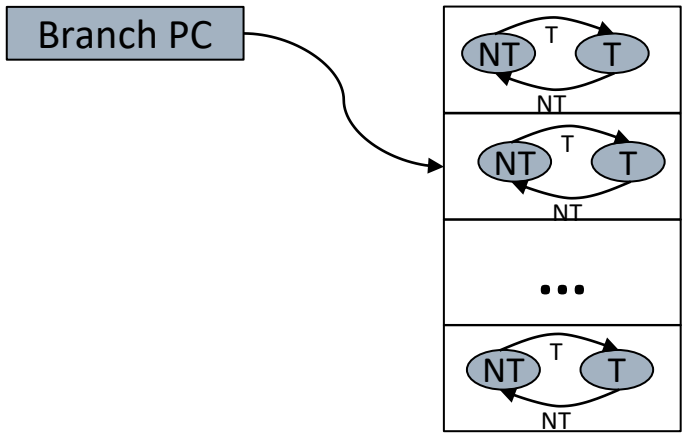
```
       lw   0  1  cnt1
       lw   0  2  cnt2
       lw   0  3  one
outer  noop
       beq  0  1  exit        //B1
       noop
       lw   0  2  cnt2
inner  beq  0  2  done        //B2
       add  4  3  4
       add  2  3  2
       beq  0  0  inner       //B3
done   add  1  3  1
       beq  0  0  outer       //B4
exit   halt
cnt1   .fill 30
cnt2   .fill 2
one    .fill -1
```

Problem 3: Branch Prediction

c. How many branches are predicted correctly if we use local last time 1-bit predictor?

Local 1-bit Predictor: In this prediction scheme, the PC of a branch instruction is used to index into a table. Each table entry for a branch is a 1-bit predictor, which predicts the same outcome as last time for that branch. Assume that the entries are initialized to Not Taken.

Local Last Time Predictor Table




```
int i, j, x = 0;
for (i=0; i != 30; i++) {
    for (j=0; j != 2; j++) {
        x++;
    }
}
```

```
        lw   0  1  cnt1
        lw   0  2  cnt2
        lw   0  3  one
outer   noop
        beq  0  1  exit        //B1
        noop
        lw   0  2  cnt2
inner   beq  0  2  done        //B2
        add  4  3  4
        add  2  3  2
        beq  0  0  inner       //B3
done    add  1  3  1
        beq  0  0  outer       //B4
exit    halt
cnt1    .fill 30
cnt2    .fill 2
one     .fill -1
```


Total executions:	B1: 31	B2: 90	B3: 60	B4: 30
NT executions:	B1: 30	B2: 60	B3: 0	B4: 0
Correct predictions:	**B1: 30**	**B2: 31**	**B3: 59**	**B4: 29**

For More Depth…

- Lecture 15_3 – Branch prediction

- Test your knowledge:
  - Winter 2019, Problem 2

Problem 4: Multi-level Page Tables

You need to design a 3-level hierarchical page table. Physical memory is with 8KB (byte-addressable); virtual memory addresses are 17-bits long. A page is 32 bytes & each page table entry is 2bytes. The 1st level table & each of the 2nd level & 3rd level page tables require precisely one page of storage.

a. Determine the bit positions for the following fields in a virtual address and a physical address.

Virtual Address			
1st level offset	2nd level offset	3rd level offset	Page offset
Bits:	Bits:	Bits:	Bits:

Physical Address	
Physical Page Number	Page offset
Bits:	Bits:

Problem 4: Multi-level Page Tables

You need to design a 3-level hierarchical page table. Physical memory is with 8KB (byte-addressable); virtual memory addresses are 17-bits long. A page is 32 bytes & each page table entry is 2bytes. The 1st level table & each of the 2nd level & 3rd level page tables require precisely one page of storage.

a. Determine the bit positions for the following fields in a virtual address and a physical address.

Virtual Address			
1st level offset	2nd level offset	3rd level offset	Page offset
Bits:16-13	Bits:12-9	Bits:8-5	Bits:4-0

Physical Address	
Physical Page Number	Page offset
Bits:12-5	Bits:4-0

Problem 4: Multi-level Page Tables

b. Assume that at the beginning of a program's execution only the 1st level table is in main memory. For the program shown below, calculate the total size of all page tables (including the 1st level, 2nd level and 3rd level page tables) in main memory after executing the `for` loop. Assume the array `BigArray` is allocated at the virtual address 0x0 and '`char`' = 1 byte. Show your work.


```
char BigArray[2048*32];
for (i = 0; i < 2048; i += 16){
        sum += BigArray[i*32];
}
```


Total 2nd Level Page Tables?

Total 3rd Level Page Tables?

Total Size in memory?

Problem 4: Multi-level Page Tables

b.
```
char BigArray[2048*32];

    for (i = 0; i < 2048; i += 16){
        sum += BigArray[i*32];
    }
```

Problem 4: Multi-level Page Tables

b.
```
char BigArray[2048*32];
      for (i = 0; i < 2048; i += 16){
            sum += BigArray[i*32];
      }
```


Total 2nd Level Page Tables? 8

Total 3rd Level Page Tables? 128

Total Size in memory? (1+8+128)*32Bytes = 4384Bytes

For More Depth…

- Video Review #12 – Virtual Memory Overview
  - https://www.youtube.com/watch?v=2tMaN-z76EY
  - https://www.eecs.umich.edu/courses/eecs370/eecs370.f20/video_reviews/

- Test your knowledge:
  - Fall 2019, Problem 5

Best of Luck!

- If you liked EECS 370, you may also like 470, 482, 483

- Best of luck with all your studies!

Logistics

- There are 3 videos for lecture 26
  - L26_1 – Final_Exam-Review
  - L26_2 – Final_Exam-Review-Questions_1C+VM
  - L26_3 – Final_Exam-Review-Questions_BP+MLPT

- There is no worksheet for lecture 14
  1. The worksheets are the past exams!
  2. https://www.eecs.umich.edu/courses/eecs370/eecs370.f20/exams/