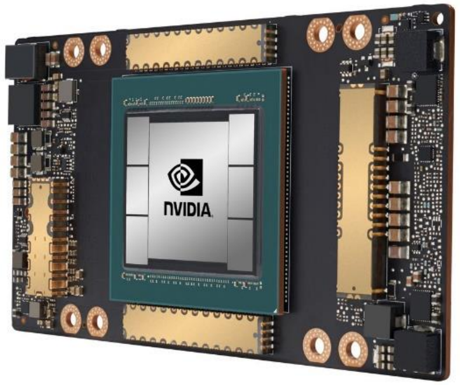
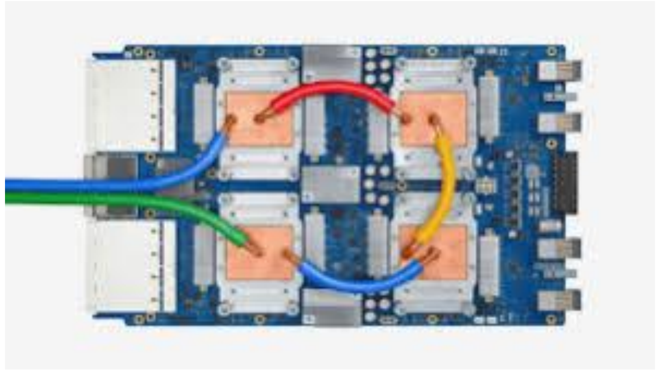
CSCB58:
Computer Organization



Prof. Gennady Pekhimenko

University of Toronto

Fall 2020



The content of this lecture is adapted from the lectures of Larry Zheng and Steve Engels

CSCB58 Week 12

Logistics

- Exam details
  - 90 mins over Quercus
  - Multiple choice, answers typed in, and writing on the paper -> upload image
  - Selective oral verification (after the exam) to avoid plagiarism

Resources

- We ask that you restrict yourself to:
  - Materials linked from the course website (slides, reading guides, the textbook)
  - Our Zoom chat (for clarifications or if you have technical problems)
- In particular, please don't:
  - Frantically google or search stack overflow (you shouldn't need them)
- Discuss the exam with anyone (except your cat, if you must)

Types of questions

- It’s a Quercus quiz, similar to what we had already (just longer)
- In addition to conceptual questions, we could ask you to:
  - Analyze a combinational circuit (including transistor diagrams)
  - Design a combinational circuit
  - Analyze a sequential circuit (or waveform)
  - Design a sequential circuit or FSM

Types of questions (2)

- Set signals on processor datapath
  - i.e., explain how an operation is performed on the processor
- Translate assembly to machine code and vice versa
- Write assembly code according to requirement
- Translate between assembly and pseudocode (C-like)
- **Count hits in a cache**
- Calculate throughput on a pipelined processor

How to study for final exam

1. Review lecture slides
2. Review what you did for labs
3. Review quizzes
4. Whenever confused, ask on piazza or come to office hours

Exam tips

- Pay attention to details
  - Don't just skim over the slides and labs and "get the idea".
  - Find a way to test yourself …
- The quiz questions are the best examples of the questions we'll use.
- The questions in the textbook are the next best.
- Identify a process for solving each kind of problem.
  - Ex: for assembly language questions, write in pseudocode and translate
  - Ex: for design questions, create a truth table, then go to a k-map

Let’s do some practice

Cache Parameters

- 128 B cache, 12-bit address, direct-mapped, 32 B blocks
- #offset bits = log2(32) = 5
- #sets = 128/32 = 4 sets
- #set/index bits = log2(4) = 2
- #tags bits = 12 - 5 – 2 = 5

Cache Parameters (example)

- Example: 0x600
- In binary: 0000 0110 0000

- 0000 0|11|0 0000

  tag idx offset

Cache Operation

Addr	Tag	Set	Offset	1st	2nd
0x070	00000	11	10000	M	H
0x080	00001	00	00000	M	M
0x068	00000	11	01000	H	H
0x190	00011	00	10000	M	M
0x084	00001	00	00100	M	M
0x178	00010	11	11000	M	M
0x08C	00001	00	01100	H	H
0xF00	11110	00	00000	M	M
0x064	00000	11	00100	M	M

Set	V	Tag
00		
01		
10		
11		

Cache Parameters - 2

- 128 B cache, 12-bit address, 2-way/LRU, 32 B blocks
- #offset bits = log2(32) = 5
- #sets = 128/(32*2) = 2 sets
- #set/index bits = log2(2) = 1
- #tags bits = 12 - 5 – 2 = 6

Cache Operation (yet again ;)

Addr	Tag	Set	Offset	1st	2nd
0x070	000001	1	10000	M	H
0x080	000010	0	00000	M	H
0x068	000001	1	01000	H	H
0x190	000110	0	10000	M	M
0x084	000010	0	00100	H	H
0x178	000101	1	11000	M	H
0x08C	000010	0	01100	H	H
0xF00	111100	0	00000	M	M
0x064	000001	1	00100	H	H

		Way 1		*Way 0*	
Set	LRU	V	Tag	V	Tag
0					
1					

K-Maps -> Truth table

Y	$\bar{C}\bar{D}$	$\bar{C}D$	$C\bar{D}$	CD
$\bar{A}\bar{B}$	1	1	0	1
$\bar{A}B$	0	1	0	0
$A\bar{B}$	0	X	X	X
AB	1	0	1	1

Fill in the truth table on the right, given the Karnaugh map values above.

A	B	C	D	Y
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

K-Maps -> Truth table

Y	$\bar{C}\bar{D}$	$\bar{C}D$	$C\bar{D}$	CD
$\bar{A}\bar{B}$	1	1	0	1
$\bar{A}B$	0	1	0	0
$A\bar{B}$	0	X	X	X
AB	1	0	1	1

Fill in the truth table on the right, given the Karnaugh map values above.

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

K-Maps -> Truth table

Y	$\bar{C}\bar{D}$	$\bar{C}D$	$C\bar{D}$	CD
$\bar{A}\bar{B}$	1	1	0	1
$\bar{A}B$	0	1	0	0
$A\bar{B}$	0	X	X	X
AB	1	0	1	1

Y	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	0	1
$\bar{A}B$	0	1	0	0
AB	0	X	X	X
$A\bar{B}$	1	0	1	1

On the Karnaugh map above, draw the minterm groupings that would result in the most optimized circuit possible.

K-Maps -> Truth table

Y	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	1	0	1
$\overline{A}B$	0	1	0	0
AB	0	X	X	X
$A\overline{B}$	1	0	1	1

Y = B’D’ + A’C’D + AC

Write the Boolean equation for the optimized groupings.

Binary numbers: add/sub

- Perform 10 – 17 in 6-bit binary numbers using 2’s complement.
- Show each step of converting decimal to binary, converting to 2’s complement, performing the subtraction, then converting the result back to decimal.

- 001010 - 010001 =
- = 001010 + 101111
- = 111001
- = -7

Decrypt assembly

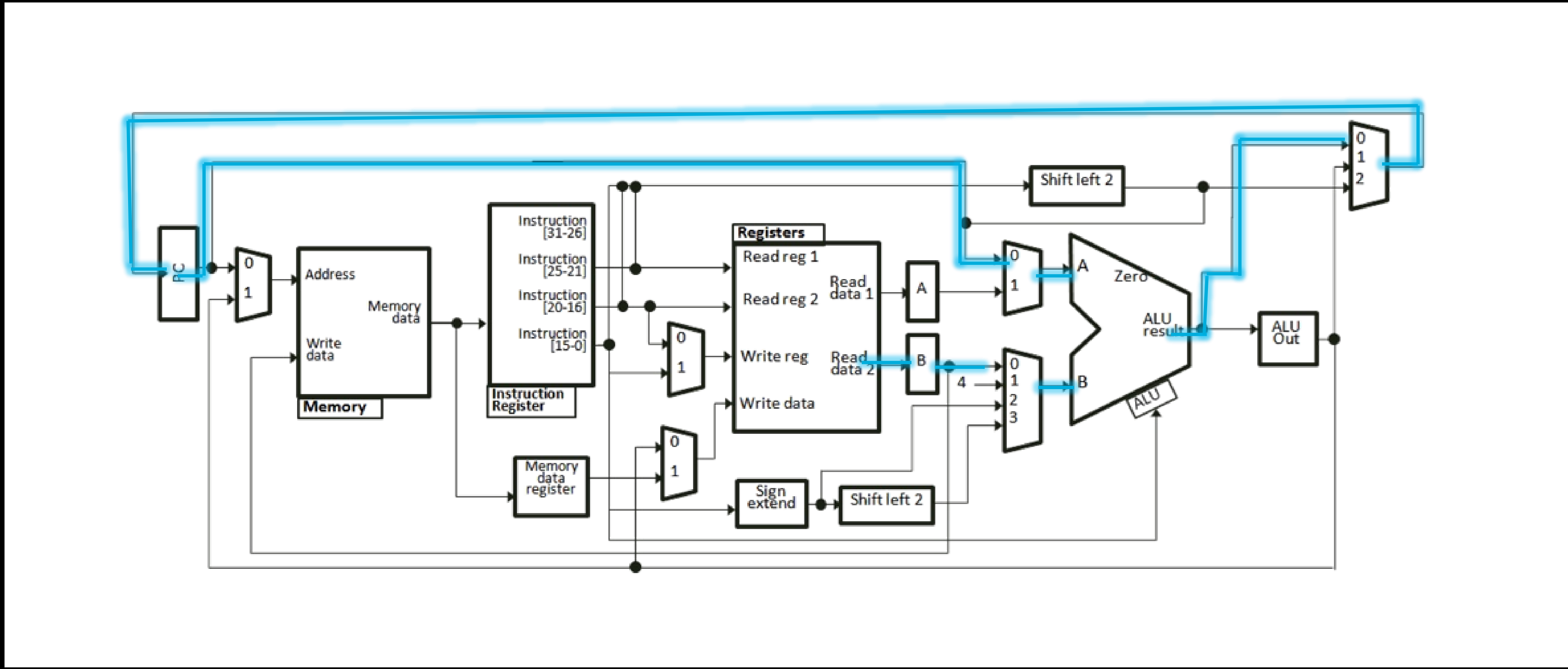
```
            .data
len:        .word     5
list:       .word     -4, 6, 7, -2, 1
            .text
main:       la $s1, len
            lw $t1, 0($s1)
            addi $t1, $t1, -1
            la $s0, list
            lw $t2, 0($s0)
alpha:      addi $t1, $t1, -1
            addi $s0, $s0, 4
            lw $t0, 0($s0)
            sub $t3, $t2, $t0
            blez $t3, beta
            add $t2, $t0, $zero
beta:       bgtz $t1, alpha
            add $v0, $zero, $t2
            jr $ra
```

Write the operation performed by an assembly program

Return the minimum item of an array

Datapath

Consider the datapaths below, highlight the path that the data needs to take, from start to finish:
Increment the program counter by the value in $so.

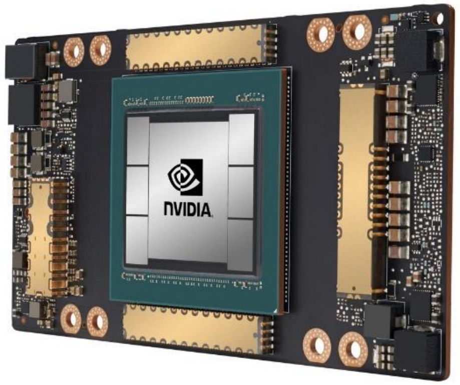


Small Assembly

push $t: Push the value in $t onto the stack

```
addi   $sp, $sp, -4
sw     $t, 0($sp)
```

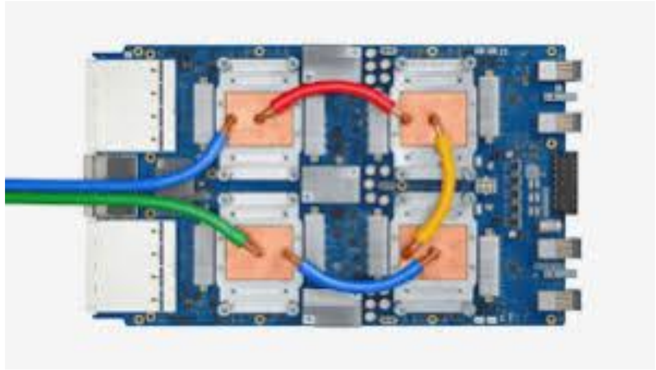
CSCB58: Computer Organization



Prof. Gennady Pekhimenko

University of Toronto

Fall 2020



The content of this lecture is adapted from the lectures of Larry Zheng and Steve Engels