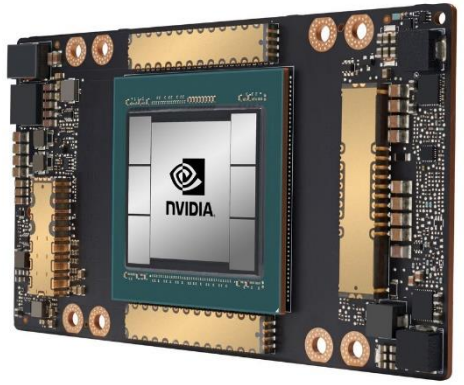
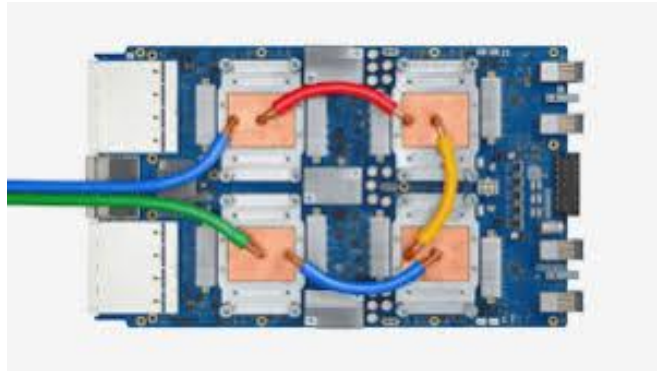
CSCB58:
Computer Organization



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 10: Summary

Week 10 Summary

It is all about Assembly:

- Basic instructions
  - Decoding
  - Interpretation

Question #1

- What are the following assembly language instructions doing?

```
sub $t7, $t0, $t1
```

Subtract register `$t1` from `$t0` and placing the result into `$t7`

```
andi $t7, $t0, 15
```

Bitwise AND between register `$t0` and 15 (`1111`), with the result placed into register `$t7`

```
sra $t2, $t1, 2
```

Arithmetic shift of register `$t1` two bits to the right, with the result stored in `$t2`

As a reminder…

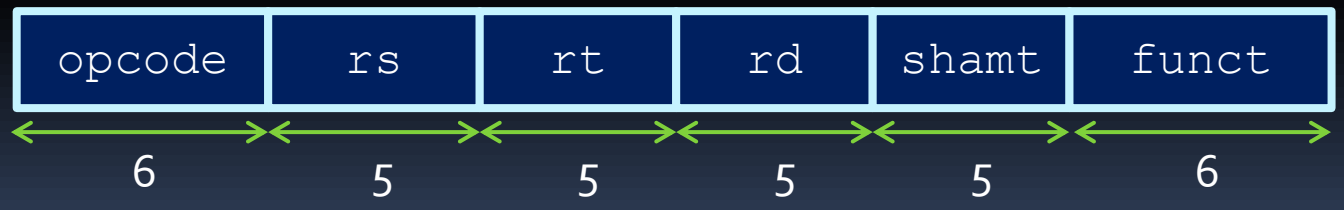
- MIPS register values:
  - Register 0 ($zero): value 0 -- always.
  - Register 1 ($at): reserved for the assembler.
  - Registers 2–3 ($v0, $v1): return values
  - Registers 4–7 ($a0-$a3): function arguments
  - Registers 8–15, 24–25 ($t0-$t9): temporaries
  - Registers 16–23 ($s0-$s7): saved temporaries
  - Registers 28–31 ($gp, $sp, $fp, $ra): memory and function support
  - Registers 27–28: reserved for OS kernel

Question #2

- How do you translate the following assembly language instruction into machine code?

```
add $t7, $t0, $t1
```

R-type instruction!



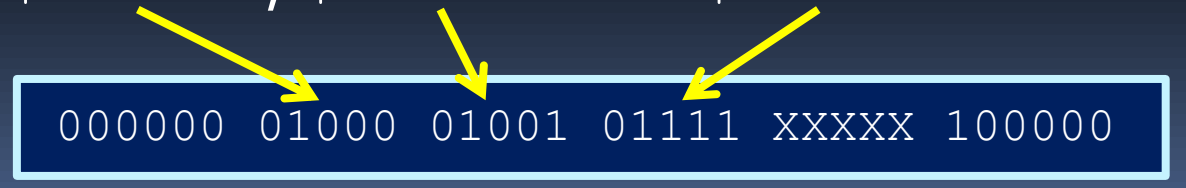
Question #2

```
add $t7, $t0, $t1
```

- Step #1: The opcode
  - Arithmetic operations start with six 0's, and have the function identifier at the end.

```
000000 sssss ttttt ddddd XXXXX 100000
```

- Step #2: The register values
  - Remember that $t0 does not translate to register 0
  - The temporary registers start at register 8, so $t0 → 8, $t1 → 9 and $t7 →15



Question #3

- What are the following assembly language instructions doing?

```
beq $t2, $zero, top
```

Jump to the line with label "top" if register `$t2` is equal to `0` (`$zero`)

```
jalr $t0
```

Store the current PC location into `$ra` (register `$31`) and jump to the location stored in register `$t0`

Question #4

- How do you translate the following assembly language instruction into machine code?

```
xori $t7, $t0, -1
```

I-type instruction!

Question #4

```
xori $t7, $t0, -1
```

- Step #1: The opcode
  - I-type instructions start with the opcode value:

```
001110 sssss ttttt iiiiiiiiiiiiiiii
```

- Step #2: The register values
  - Register `$t0` translates to register `8`, and register `$t7` translates to register `15`
  - 16-bit immediate value is `-1`.

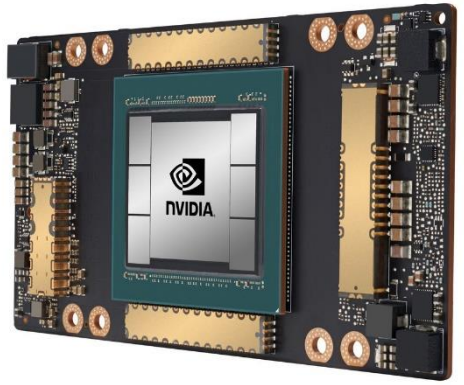
```
001110 01000 01111 1111111111111111
```

Question #5

- How do you write an assembly language program that can swap the values in `$t0` and `$t1`, using `$t2` as a temp value?

```
add $t2, $zero, $t0
add $t0, $zero, $t1
add $t1, $zero, $t2
```

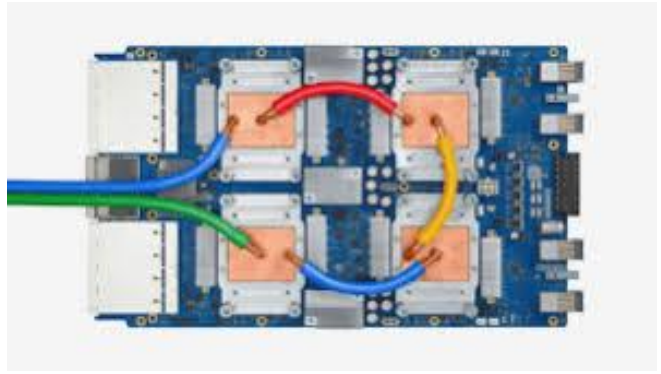
CSCB58:
Computer Organization



Prof. Gennady Pekhimenko

University of Toronto

Fall 2020



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