

**Objectives:**

- Computers are not magic
- CPUs and the fetch-execute cycle

2.

Q1: What number is represented by the bit pattern 010001?

Q2: What number is represented by the bit pattern 100011?

Q3: What number is represented by the bit pattern 011111?

### 3. Using 8 bits, **write these decimal numbers in binary representation:**

$$3_{10} =$$

$$10_{10} =$$

$$67_{10} =$$

$$7_{10} =$$

$$14_{10} =$$

$$254_{10} =$$

### 1. Computer Science Terminology - did your neighbor do the readings?

Discuss with your neighbor what a Computer Scientists means by the following terms and give an example of each:

- fetch-execute cycle
- CPU
- opcode
- operand
- register
- condition code

### 4. Arithmetic can be done with binary numbers:

<https://www.youtube.com/watch?v=GcDshWmhF4A>

$$\begin{array}{r} 11110_2 \\ + 00111_2 \\ \hline \end{array}$$

2

### 5. Computing a Quiz Average: Pseudo-code to calculate a quiz average

1. get number of quizzes
2. sum := 0
3. count := 0
4. while count < number of quizzes
  - get quiz grade
  - sum = sum + quiz grade
  - count = count + 1
5. average = sum / number of quizzes
6. display average

### 6. Write pseudo-code to print the highest quiz score:

A simple machine language:

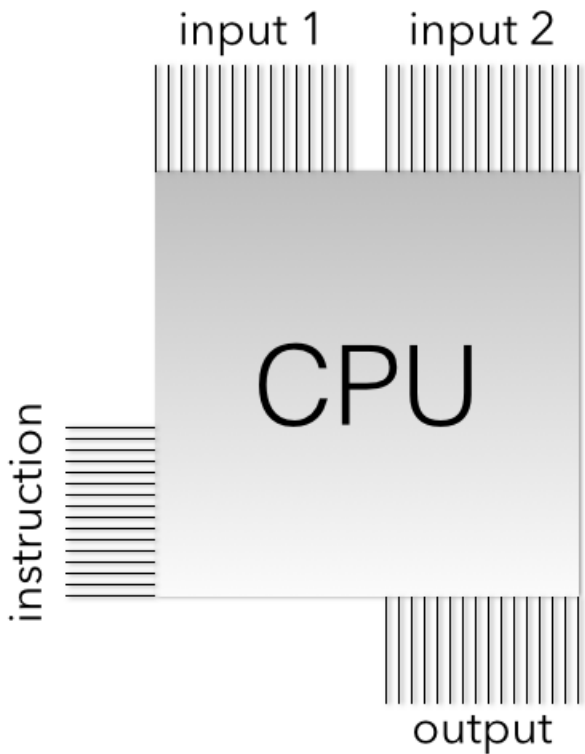
- ZERO\_REG DESTR** - puts a zero in the specified register.
- ADD SRCR1 + SRCR2 -> DESTR** - add two registers together and write the result in the destination register.
- ADD SRCR1 + CONSTANT -> DESTR** - add a register to a constant and write the result in the destination register.
- SUB SRCR1 - SRCR2 -> DESTR** - subtract one register from another and write the result in the destination register.
- SUB SRCR1 - CONSTANT -> DESTR** - subtract a constant from a register and write the result in the destination register.
- LOAD DESTR <- [BASER + CONSTANT]** - add the value of a register to a constant to compute a memory address and copy 4 bytes starting at that address to the destination register.
- STORE SRCR1 -> [BASER + CONSTANT]** - add the value of a register to a constant to compute a memory address and copy the source register to 4 bytes of memory starting at that address.
- BR.\_\_\_\_ PCOFFSET** - the branch instruction specifies a combination of condition codes (n, z, p); if any of the specified condition codes holds a 1, the PC is set to  $PC + 2 + 2(PCOFFSET)$ . Otherwise PC is set to  $PC + 2$ .

For all instructions other than the branch, PC is set to  $PC + 2$ . Any instruction that writes a general-purpose register also set the condition code bits: if new value is negative then  $n=1$ , else  $n=0$ ; if new value is zero then  $z=1$ , else  $z=0$ ; if new value is positive then  $p=1$ , else  $p=0$ .

7. Decoding an instruction:

|                       | 15   | 14 | 13 | 12 | 11 | 10 | 9 | 8         | 7 | 6 | 5       | 4         | 3 | 2   | 1 | 0 |
|-----------------------|------|----|----|----|----|----|---|-----------|---|---|---------|-----------|---|-----|---|---|
| ADD <sup>+</sup>      | 0001 |    |    |    | DR |    |   | SR1       |   |   | 0       | 00        |   | SR2 |   |   |
| ADD <sup>+</sup>      | 0001 |    |    |    | DR |    |   | SR1       |   |   | 1       | constant5 |   |     |   |   |
| ZERO_REG <sup>+</sup> | 0101 |    |    |    | DR |    |   | 0         | 0 | 0 | 1       | 0         | 0 | 0   | 0 | 0 |
| BR                    | 0000 |    |    |    | n  | z  | p | PCOffset9 |   |   |         |           |   |     |   |   |
| LOAD <sup>+</sup>     | 0110 |    |    |    | DR |    |   | BaseR     |   |   | offset6 |           |   |     |   |   |
| STORE                 | 0111 |    |    |    | SR |    |   | BaseR     |   |   | offset6 |           |   |     |   |   |
| SUB <sup>+</sup>      | 0001 |    |    |    | DR |    |   | SR1       |   |   | 0       | 00        |   | SR2 |   |   |
| SUB <sup>+</sup>      | 0001 |    |    |    | DR |    |   | SR1       |   |   | 1       | constant5 |   |     |   |   |

8. Schematic of a CPU:



9. Decoding 16 bit-string instructions:

- 0111011001000100
- 0001100011000010
- 0101101000100000
- 0000110000001100

11. Execute a machine code program - me:

r0

r1

r2

r3

r4

r5

r6

r7

PC

N

Z

P

address

a0

a4

a100

a102

a104

a106

a108

a110

7

zero\_reg r1

load r2 <- [r1+0]

br.zp 1

sub r1 - r2 -> r2

store r2 <- [r1+4]

// done

12. Execute a machine code program - you:

r0

r1

r2

r3

r4

r5

r6

r7

PC

N

Z

P

address

a0

a4

a100

a102

a104

a106

a108

a110

-3

zero\_reg r1

load r2 <- [r1+0]

br.zp 1

sub r1 - r2 -> r2

store r2 <- [r1+4]

// done

13. What does this code do?

10. All about that branch, 'bout that branch, 'bout that branch ...

PC

122

Condition codes

N

1

Z

0

P

0

BR.N 6

PC =

BR.NZ -6

PC =

PC

108

Condition codes

N

0

Z

1

P

0

BR.NZP 22

PC =

BR.ZP -10

PC =

Workspace:

14. Execute a machine code program - you:

r0

r1

r2

r3

r4

r5

r6

r7

PC

N

Z

P

100

address

a0

a4

a8

3

10

a100

a102

a104

a106

a108

a110

a112

a114

a116

a118

a120

a122

zero\_reg r1

store r1 -> [r1+8]

load r2 <- [r1+0]

br.nz 7

sub r2 - 1 -> r2

store r2 -> [r1+0]

load r3 <- [r1+4]

load r4 <- [r1+8]

add r3 + r4 -> r4

store r4 -> [r1+8]

br.pnz -9

// done

15. What does this code do?