

# 09 – MACHINE LANGUAGE

COMP256 – COMPUTING ABSTRACTIONS

DICKINSON COLLEGE

## MACHINE LANGUAGE

- A machine language is created by defining patterns of bits to correspond to common operations (e.g. moving data, addition, etc.).
- The control unit handles the translation. Its circuitry will:
  - fetch a machine language instruction from main memory,
  - decode it,
  - and configure the machine to execute it.
- Each processor has its own unique machine language.

## RAISING THE LEVEL OF ABSTRACTION

- Micro-program instructions require thinking about the internal machine details and how they need to be configured to carry out a desired operation.
  - Complex distraction from the task at hand
  - Possible to specify meaningless micro-instructions
- A higher level of abstraction (e.g. machine language) allows programmers to focus on the desired operations (e.g. move data, add, etc...) without worrying about details such as switch positions.
  - Increases efficiency
  - Reduces mistakes

## MACHINE LANGUAGE INSTRUCTIONS

- Machine language instructions have two parts:
  - The OpCode (operation code) specifies the operation to be performed.
  - The Operands (arguments) specify the data to be used in the operation.
- The format of a machine language instruction specifies how its bits encode the opcode and its operands.
  - For Example three of the instruction formats for the Knob & Switch computer are:

OpCode (10 bits)	Operands (6 bits)
OpCode (9 bits)	Operands (7 bits)
OpCode (16 bits)	

## KNOB & SWITCH MACHINE LANGUAGE

- Data Movement Instructions

1000 0001 0	RegC(2) address (5)
1000 0010 0	RegC(2) address (5)
1001 0001 0000	RegA (2) RegC (2)

$RegC = MM[address]$   
 $MM[address] = RegC$   
 $RegC = RegA$

## KNOB & SWITCH MACHINE LANGUAGE

- Other Instructions:

0000 0000 0000 0000

No Operation

1111 1111 1111 1111

Halt

- All Knob & Switch programs should end in with a Halt instruction.

## KNOB & SWITCH MACHINE LANGUAGE

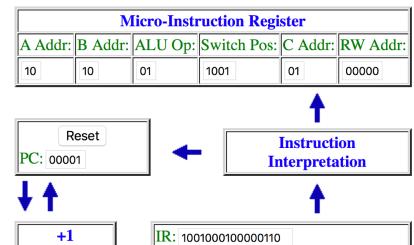
- Arithmetic and Logic Instructions:

1010 0001 00	RegC (2) RegA(2) RefB(2)
1010 0010 00	RegC (2) RegA(2) RegB(2)
1010 0011 00	RegC (2) RegA(2) RegB(2)
1010 0100 00	RegC (2) RegA(2) RegB(2)

$RegC = RegA + RegB$   
 $RegC = RegA - RegB$   
 $RegC = RegA \& RegB$   
 $RegC = RegA | RegB$

## THE CONTROL UNIT

- The control unit directs the fetch, decode execute cycle:
  - The Program Counter (PC) hold the memory address of the next instruction to fetch.
  - The Instruction Register (IR) holds the instruction currently being decoded and executed.
- The instruction interpretation circuitry decodes the machine language instruction into a micro-instruction that causes the execution of the operation.



## ACTIVITY

- Give a machine language program (i.e. a sequence of machine language instructions) that accomplish the following operation and then stop:
  - $MM[10] = MM[11] + MM[12]$

**Pro Tip:** Always be sure to reset the PC to 0 before running your program!



PC: 00001

## KNOB & SWITCH MACHINE LANGUAGE

- Branching Instructions:

0000 0001 000

address (5)

PC = address

0000 0010 000

address (5)

If ALU Zero Flag is set then  
PC = address

else

PC = PC + 1

0000 0011 000

address (5)

If ALU Negative Flag is set then  
PC = address

else

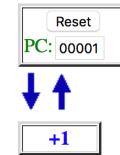
PC = PC + 1

## BRANCHING INSTRUCTIONS

- Branching instructions provide the mechanism for programs to make decisions.

- Branching instructions change the address from which the next instruction will be fetched by changing the Program Counter (PC)

- Branches are based upon the ALU Flags from the most recent operation (e.g. zero result, or negative result).



A: 0000000000000000	B: 0000000000000000
Arithmetic and Logic Unit (ALU)	
C: 0000000000000000	ALU Operation
<input checked="" type="checkbox"/> Zero	A+B
<input type="checkbox"/> Negative	A-B
<input type="checkbox"/> Unsigned Overflow	A <b>⊕</b> B
<input type="checkbox"/> Signed Overflow	A <b>⊖</b> B

## ACTIVITIES

- Write machine language programs that do the following:

- If  $mm[20] < 0$  then  $MM[20] = 0$

- Note that this is similar to the high level language:  
  - if  $x < 0$  then  $x = 0$

- If  $mm[20] == 0$  then  $MM[21] = MM[22]$ , else  $MM[21] = MM[23]$

- Note that this is similar to the high level language:  
  - if  $x == 0$ ,  $y = a$  else  $y = b$