MARIE Simulator
Schedule

- **Today** – MARIE assembly programming
- **Friday 17th** – MARIE assembly programming
- **Monday 20th** – No class
- **Wednesday 22nd**
  - MARIE assembly programming
  - **Quiz 3!**
    - Topic: Assembly programming!
    - *I will give you Table 4.7 from the book...*
Prelab Setup – MARIE Simulator

- If you are using your own laptop, make sure Java is installed
  - http://java.oracle.com (Java SE, then download Java 7 JRE)

- Get the MARIE simulator now
  - ECPE 170 Sakai site under “Resources”
  - or Textbook website:

- Today’s goals:
  - Run some sample programs
  - And write your own!
# Recap – MARIE Instructions (Full)

<table>
<thead>
<tr>
<th>Binary</th>
<th>Hex</th>
<th>Instruction</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>1</td>
<td>LOAD X</td>
<td>Load contents of address X into AC</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>STORE X</td>
<td>Store contents of AC at address X</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>ADD X</td>
<td>Add contents of address X to AC</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>SUBT X</td>
<td>Subtract contents of address X from AC</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>INPUT</td>
<td>Input value from keyboard into AC</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>OUTPUT</td>
<td>Output value in AC to display</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>HALT</td>
<td>Terminate program</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>SKIPCOND</td>
<td>Skip next instruction on condition based on AC value</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>JUMP X</td>
<td>Load value of X into PC</td>
</tr>
<tr>
<td>1010</td>
<td>A</td>
<td>CLEAR</td>
<td>Set AC to 0</td>
</tr>
<tr>
<td>1011</td>
<td>B</td>
<td>ADDI X</td>
<td>Add contents of address Mem[X] to AC</td>
</tr>
<tr>
<td>1100</td>
<td>C</td>
<td>JUMPI X</td>
<td>Load contents of address Mem[X] into PC</td>
</tr>
<tr>
<td>1101</td>
<td>D</td>
<td>LOADI X</td>
<td>Load contents of address Mem[X] into AC</td>
</tr>
<tr>
<td>1110</td>
<td>E</td>
<td>STOREI X</td>
<td>Store contents of AC at address Mem[X]</td>
</tr>
</tbody>
</table>

See table 4.7 in book!
High-Level Language

\[ \begin{align*}
X &= 5 \\
Y &= 7 \\
Z &= X + 7
\end{align*} \]

Assembly

\[ \begin{align*}
&\text{LOAD } X \\
&\text{ADD } Y \\
&\text{STORE } Z \\
&\text{DEC } X, 5 \\
&\text{DEC } Y, 7 \\
&\text{DEC } Z, 0
\end{align*} \]
Using the MARIE Simulator

implement

To use the simulator

1. Unzip the downloaded archive into a folder on your U: drive
2. Browse the files and locate MarieSim.jar

MarieSim is a JAVA application

- Unless your computer has .JAR files associated with the Java machine, you will need to run the program “by hand”
  - Go to Start Menu, pick “Run…”
  - Enter: java -jar MarieSim.jar
### MARIE Simulator

**AC**: 0000 (Hex)
**IR**: 0000 (Hex)
**MAR**: 000 (Hex)
**MBR**: 0000 (Hex)
**PC**: 000 (Hex)

**Ready to load program instructions.**

<table>
<thead>
<tr>
<th>Hex</th>
<th>000</th>
<th>010</th>
<th>020</th>
<th>030</th>
<th>040</th>
<th>050</th>
<th>060</th>
<th>070</th>
<th>080</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>2</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>3</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>4</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>5</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>6</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>7</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>8</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>9</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>A</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>B</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>C</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>D</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>E</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>F</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
</tbody>
</table>
Using the MARIE Simulator

- Our programs are written in MARIE assembly language
  - “.mas” files

- Need to use the assembler before running (simulating) the program!
  - What does the assembler do again?

- To start, do “File → Edit”
  - Opens editor
  - Type in your file, or “File → Open” to load
  - Choose file “Ex4_1.mas”
ORG 100  / Example 4.1
Load  Addr  /Load address of first number to be added
Store  Next  /Store this address is our Next pointer
Load  Num  /Load the number of items to be added
Subt  One  /Decrement
Store  Ctr  /Store this value in Ctr to control looping
Load  Sum  /Load the Sum into AC
AddI  Next  /Add the value pointed to by location Next
Store  Sum  /Store this sum
Load  Next  /Load Next
Add  One  /Increment by one to point to next address
Store  Next  /Store in our pointer Next
Load  Ctr  /Load the loop control variable
Subt  One  /Subtract one from the loop control variable
Store  Ctr  /Store this new value in loop control variable
Skipcond 000  /If control variable < 0, skip next instruction
Jump  Loop  /Otherwise, go to Loop
Halt  /Terminate program

Addr,  Hex  117  /Numbers to be summed start at location 118
Next,  Hex  0  /A pointer to the next number to add
Num,  Dec  5  /The number of values to add
Sum,  Dec  0  /The sum
Ctr,  Hex  0  /The loop control variable
One,  Dec  1  /Used to increment and decrement by 1
Dec  10  /The values to be added together
Dec  15
Dec  20
Dec  25
Dec  30
Using the MARIE Simulator

Assembly file format:

- **Labels**: define addresses we want to access
  - End with a comma (,)
- **Opcode**: the operation to perform
- **Operands**: other data needed by the instruction
- **Comments**: you know, comments
  - Comments start with / symbol in MARIE

Typical MARIE line: *(Label is optional)*

```
Label, opcode operands / comments
```
What is DEC? HEX? ORG? END?

Are they assembly commands for the processor?

No – these are commands for the assembler only!

DEC X – The number to follow is written in base 10 (please convert to binary)

HEX X – The number to follow is written in base 16 (please convert to binary)

ORG X – Please store this program in memory starting at memory address X (in Hex)

END - Stop Assembly (finished!)
Using the MARIE Simulator

- Ready to run simulator?
- Assemble source code
  - "Assemble → Assemble Current File"
- Files produced by assembler
  - .lst file = Original assembly code + machine code
  - .map file = Symbol table from assembly process
  - .mex file = Machine code (only)
- Errors? Listing file will indicate line and problem
- No errors? Ready to simulate!
Assembly listing for: Ex4_2.mas
Assembled: Mon Oct 03 10:37:06 PDT 2011

/ Example 4.1

ORG 100

100 ?10C  If  LOADX X  /Load the first value

**** Instruction not recognized.

101 410D  SUBT Y  /Subtract the value of Y, store result in AC
102 8400  SKIPCOND 400  /If AC=0, skip the next instruction
103 9108  JUMP Else  /Jump to Else part if AC is not equal to 0
104 110C  Then  LOAD X  /Reload X so it can be doubled
105 310C  ADD X  /Double X
106 210C  STORE X  /Store the new value
107 910B  JUMP Endif  /Skip over the false, or else, part to end of if
108 110D  Else  LOAD Y  /Start the else part by loading Y
109 410C  SUBT X  /Subtract X from Y
10A 210D  STORE Y  /Store Y-X in Y
10B 7000  Endif  HALT  /Terminate program (it doesn't do much!)
10C 000C  X  DEC 12  /Load the loop control variable
10D 0014  Y  DEC 20  /Subtract one from the loop control variable
END

1 error found. Assembly unsuccessful.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Defined</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Else</td>
<td>108</td>
<td>103</td>
</tr>
<tr>
<td>Endif</td>
<td>10B</td>
<td>107</td>
</tr>
<tr>
<td>If</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Then</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>10C</td>
<td>100, 104, 105, 106, 109</td>
</tr>
<tr>
<td>Y</td>
<td>10D</td>
<td>101, 108, 10A</td>
</tr>
</tbody>
</table>
To simulate, “File ➔ Load”
- Pick the .mex file created by the assembler

Code shows up in upper left window
- Addresses shown on the left
- Machine code shown on the right

Registers shown in the middle

Output (from OUTPUT instruction) on right

Bottom windows shows “memory dump”
Ways to simulate

Run: run continuously until you choose “Stop” or CPU executes a HALT

Step
Choose “Run → Set stepping mode → on” first
Let's you examine one instruction at a time

Breakpoints
Let's you pick instructions to stop at
Check the box next to the instructions’ address
Choose “Breakpoints → Run to Breakpoints”
With a partner *(if desired)*, take 5 minutes to:

- **Examine** the assembly code in the file Ex4_1.mas (already open)
- **Write the equivalent C++** (or Java, or pseudocode) for the operation being performed

Equivalent code doesn’t have to be perfect
- You could write several different C++ programs that accomplish the same tasks!
ORG 100
Load Addr /Load address of first number to be added
Store Next /Store this address is our Next pointer
Load Num /Load the number of items to be added
Subt One /Decrement
Store Ctr /Store this value in Ctr to control looping
Loop, Load Sum /Load the Sum into AC
AddI Next /Add the value pointed to by location Next
Store Sum /Store this sum
Load Next /Load Next
Add One /Increment by one to point to next address
Store Next /Store in our pointer Next
Load Ctr /Load the loop control variable
Subt One /Subtract one from the loop control variable
Store Ctr /Store this new value in loop control variable
Skipcond 000 /If control variable < 0, skip next instruction
Jump Loop /Otherwise, go to Loop
Halt /Terminate program
Addr, Hex 117 /Numbers to be summed start at location 118
Next, Hex 0 /A pointer to the next number to add
Num, Dec 5 /The number of values to add
Sum, Dec 0 /The sum
Ctr, Hex 0 /The loop control variable
One, Dec 1 /Used to increment and decrement by 1
Dec 10 /The values to be added together
Dec 15
Dec 20
Dec 25
Dec 30
int myArray[5]={10,20,30,40,50};
int num=5;
int counter=0;
int sum=0;

counter = num - 1;
do
{
    sum = sum + myArray[counter];
    counter = counter - 1;
}
while(counter >=0)
With a partner *(if desired)*, **write** and **run** a complete MARIE assembly program to accomplish the follow task:

```assembly
if X==Y then
    X = X * 2;
else
    Y = Y - X;
```

**Show me** the running program with \( X=12_{10}, Y=20_{10} \)
ORG 100

If,
LOAD   X   /Load the first value
SUBT   Y   /Subtract the value of Y, store result in AC
SKIPCOND 400  /If AC=0, skip the next instruction
JUMP   Else  /Jump to Else part if AC is not equal to 0

Then, LOAD   X   /Reload X so it can be doubled
ADD     X   /Double X
STORE   X   /Store the new value
JUMP   Endif  /Skip over the false (else) part to end of if

Else, LOAD   Y   /Start the else part by loading Y
SUBT    X   /Subtract X from Y
STORE   Y   /Store Y-X in Y
Endif, HALT   /Terminate program (it doesn't do much!)

X,     Dec 12
Y,     Dec 20
END
Homework #9 and #10
- Work individually or in teams of 2
- Each person must submit assignment!
  - Put your name and partner’s name in comments

You MUST comment your code!
- At least 90% of the lines!
- No points for uncommented code

Sakai submission
- Turn in each “.mas” source file separately
- Name them “ex428.mas”, “ex429.mas”, ...

Files should be PLAIN ASCII TEXT (use MARIE editor)
- Zero points if you give me a .doc, .docx, .pdf, etc...
- We will run your program in the simulator for grading