MARIE Simulator
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

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

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…”
(or) Run “java –jar <mariesimfile>”
Ready to load program instructions.
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**”
<table>
<thead>
<tr>
<th>Macro/mnemonic</th>
<th>Address/Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG 100</td>
<td></td>
<td>/ Example 4.1</td>
</tr>
<tr>
<td>Load</td>
<td>Addr</td>
<td>/Load address of first number to be added</td>
</tr>
<tr>
<td>Store</td>
<td>Next</td>
<td>/Store this address is our Next pointer</td>
</tr>
<tr>
<td>Load</td>
<td>Num</td>
<td>/Load the number of items to be added</td>
</tr>
<tr>
<td>Subt</td>
<td>One</td>
<td>/Decrement</td>
</tr>
<tr>
<td>Store</td>
<td>Ctr</td>
<td>/Store this value in Ctr to control looping</td>
</tr>
<tr>
<td>Load</td>
<td>Sum</td>
<td>/Load the Sum into AC</td>
</tr>
<tr>
<td>AddI</td>
<td>Next</td>
<td>/Add the value pointed to by location Next</td>
</tr>
<tr>
<td>Store</td>
<td>Sum</td>
<td>/Store this sum</td>
</tr>
<tr>
<td>Load</td>
<td>Next</td>
<td>/Load Next</td>
</tr>
<tr>
<td>Add</td>
<td>One</td>
<td>/Increment by one to point to next address</td>
</tr>
<tr>
<td>Store</td>
<td>Next</td>
<td>/Store in our pointer Next</td>
</tr>
<tr>
<td>Load</td>
<td>Ctr</td>
<td>/Load the loop control variable</td>
</tr>
<tr>
<td>Subt</td>
<td>One</td>
<td>/Subtract one from the loop control variable</td>
</tr>
<tr>
<td>Store</td>
<td>Ctr</td>
<td>/Store this new value in loop control variable</td>
</tr>
<tr>
<td>Skipcond</td>
<td>000</td>
<td>/If control variable &lt; 0, skip next instruction</td>
</tr>
<tr>
<td>Jump</td>
<td>Loop</td>
<td>/Otherwise, go to Loop</td>
</tr>
<tr>
<td>Addr, Hex</td>
<td>117</td>
<td>/Numbers to be summed start at location 118</td>
</tr>
<tr>
<td>Next, Hex</td>
<td>0</td>
<td>/A pointer to the next number to add</td>
</tr>
<tr>
<td>Num, Dec</td>
<td>5</td>
<td>/The number of values to add</td>
</tr>
<tr>
<td>Sum, Dec</td>
<td>0</td>
<td>/The sum</td>
</tr>
<tr>
<td>Ctr, Hex</td>
<td>0</td>
<td>/The loop control variable</td>
</tr>
<tr>
<td>One, Dec</td>
<td>1</td>
<td>/Used to increment and decrement by 1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>/The values to be added together</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
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
  - Start with / 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

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

1 error found. Assembly unsuccessful.

SYMBOL TABLE

<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>
Using the MARIE Simulator

- 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”
Using the MARIE Simulator

- **Ways to simulate**
  - **Run**: run continuously until you choose “Stop” or CPU executes a HALT
  - **Step**
    - Choose “Run → Set stepping mode → on” first
    - Lets you examine one instruction at a time
  - **Breakpoints**
    - Lets 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!
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)
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

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,    Dec 0   /The loop control variable
One,    Dec 1   /Used to increment and decrement by 1
Dec 10
Dec 15
Dec 20
Dec 25
Dec 30
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
Exercises 4.28 + 4.29
- Work individually or in teams of 2
- Each person must submit assignment!
- Put your name and partner’s name in comments

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

Files should be PLAIN ASCII TEXT (use NotePad or MARIE editor)
- Zero points if you give me a .doc, .docx, .pdf, scanned copy of a printout, smoke signals, etc...

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