Computer Systems and Networks

ECPE 170 – Jeff Shafer – University of the Pacific

Introduction to MARIE

Schedule

- **Today**
  - Continue with MARIE intro
  - Exam review
- **Thursday – Exam 1**
  - Chapter 2 (Data representations)
  - Chapter 3 (Digital logic)
  - Part of Chapter 4 (basic organization and memory systems. Nothing on MARIE)

- **Next Tuesday**
  - Meet in **CTC 115** for assembly programming
  - Homework 7 due

Recap – MARIE Overview

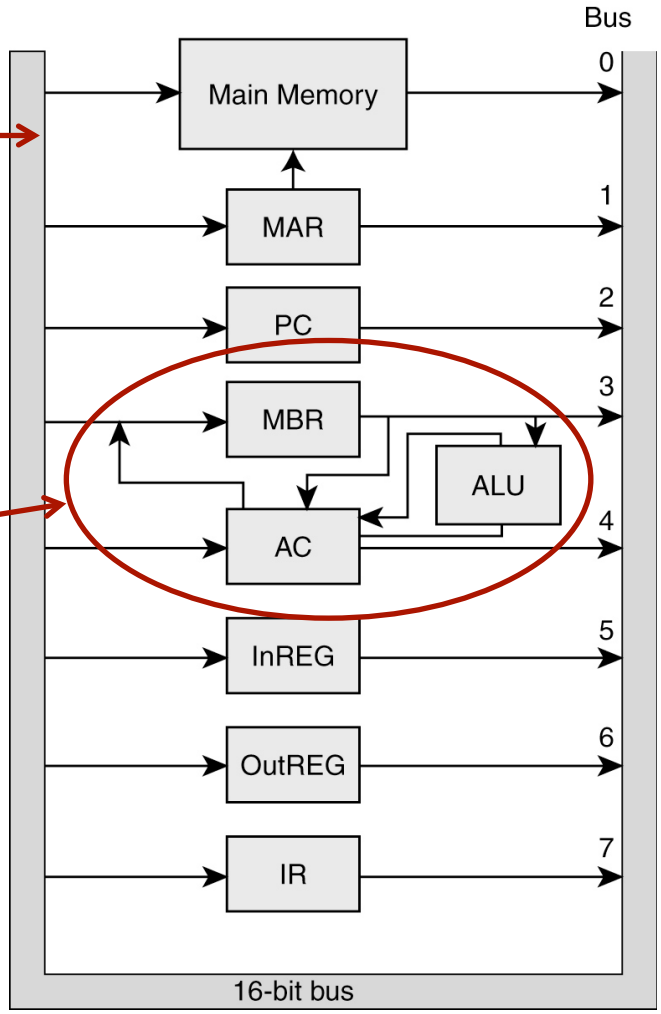
- **How does the MARIE architecture represent positive/negative numbers?**
  - Binary, two's complement data representation
- **How is MARIE's main memory configured? (# of words, size of each word)**
  - 4K words, 16 bits wide, word-addressable

Recap – MARIE Overview

- MARIE has **seven registers** for control and data movement
  - **AC?**
  - **MAR?**
  - **MBR?**
  - **PC?**
  - **IR?**
  - **InReg?**
  - **OutReg?**

Recap – MARIE Data Path

- **Common data bus**
  - Links main memory and registers
  - Each device identified by unique number
  - Bus has control lines that identify device used in operation
- **Dedicated data paths**
  - Permits data transfer between accumulator (AC), memory buffer register (MBR), and ALU without using main data bus



Recap – MARIE ISA

- **What is an Instruction Set Architecture (ISA)?**
  - Interface between hardware and software
  - Specifies the format of processor instructions
  - Specifies the primitive operations the processor can perform

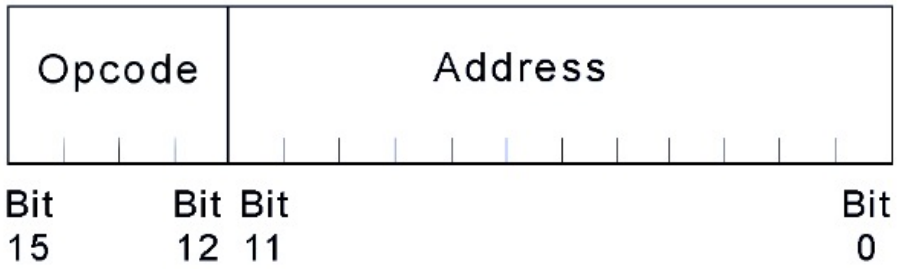
Recap – MARIE Instructions (Full)

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

See table 4.7 in book!

Recap – MARIE Instructions

- **How does MARIE format instructions in computer memory?**



- Two fields
  - **Opcode** (4 bits) – Operation code
  - **Address** (12 bits) – Address to operate to/from

Assembler

Role of Assembler

- Mnemonic instructions: LOAD 104
  - "Easy" for humans to write and understand
  - Impossible for computers to understand
- Role of **assembler**
  - Translate instructions from assembly language (*for humans*) into machine language (*for computers*)

Assembler versus Compiler

- **What's the difference between an assembler and a compiler? Which has the harder job?**
  - Assembly language → machine language
    - One-to-one correspondence
    - **Assembler** is simple!
  - High-level language → machine language
    - Many-to-one correspondence
    - **Compiler** is complicated!

Assembler Operation

- Assemblers create an **object file** *(containing machine code)* from mnemonic assembly source code in **two passes**
- Pass 1
  - Assemble as much of the program as possible
  - Builds a **symbol table** (contains memory references for all *symbols* in the program)
- Pass 2
  - Complete instructions. Fill in addresses stored in the symbol table

Assembler Operation

- Example program
  - HEX and DEC directives to specify radix of constants
- Assembler **Pass #1**
  - Create symbol table
  - Create partially-assembled instructions

Address	Instruction
100	Load X
101	Add Y
102	Store Z
103	Halt
104 X,	DEC 35
105 Y,	DEC -23
106 Z,	HEX 0000

Symbol Table: ***Name, Address***

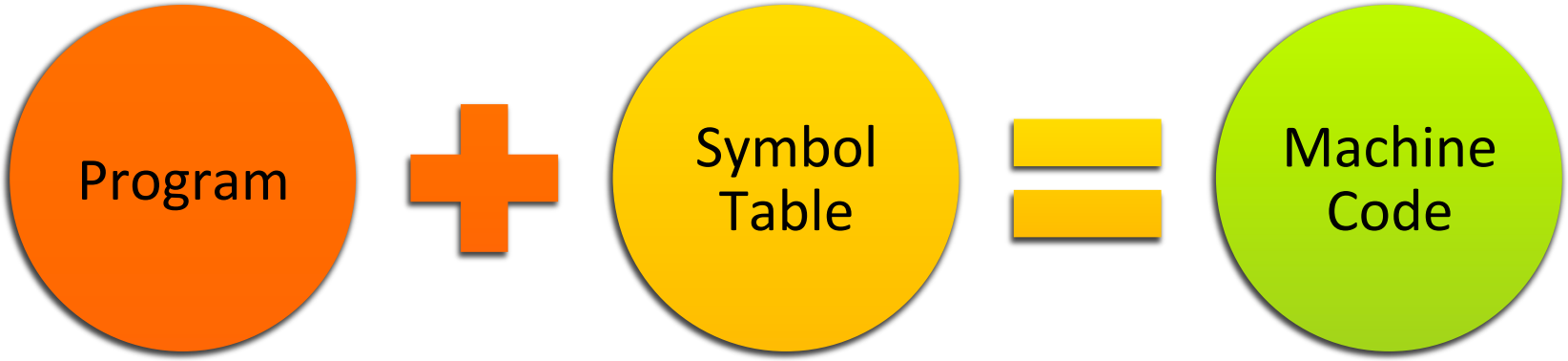
X	104
Y	105
Z	106

Partially-Assembled Program:

1	X
3	Y
2	Z
7 0 0 0	

Assembler Operation

- Assembler **Pass #2**
  - Fill in details from symbol table



Assembler Operation

Program:

Address	Instruction
100	Load X
101	Add Y
102	Store Z
103	Halt
104 X,	DEC 35
105 Y,	DEC -23
106 Z,	HEX 0000

Symbol Table:

X	104
Y	105
Z	106

Machine Code:

1 1 0 4
3 1 0 5
2 1 0 6
7 0 0 0
0 0 2 3
F F E 9
0 0 0 0

More MARIE Instructions

New Addressing Modes!

- **Direct addressing mode**
  - *All the instructions covered to date…*
  - The address of the operand is explicitly stated in the instruction
- New: **Indirect addressing mode**
  - The address of the address of the operand is given in the instruction
  - Just like **pointers** in COMP 51/53

Indirect Addressing Mode Instructions

- Four new instructions use *indirect* addressing mode: Load / store / add / jump indirect
- `LOADI X` and `STOREI X` – specified the address of the address of the operand to be loaded or stored
  - In RTL :

LOADI X

```
MAR ← X
MBR ← M[MAR]
MAR ← MBR
MBR ← M[MAR]
AC ← MBR
```

STOREI X

```
MAR ← X
MBR ← M[MAR]
MAR ← MBR
MBR ← AC
M[MAR] ← MBR
```

Indirect Addressing Mode Instructions

- `ADDI X` - Combination of `LOADI X` and `ADD X`:
  - In RTL:

ADDI X

```
MAR ← X
MBR ← M[MAR]
MAR ← MBR
MBR ← M[MAR]
AC ← AC + MBR
```

Subroutine Instructions

- Remember subroutines? (i.e. functions)
- Machine instructions can make subroutines easier to implement
  - Jump-and-store instruction (`JNS X`) provides limited subroutine functionality
  - RTL:

```
MBR ← PC
MAR ← X
M[MAR] ← MBR
MBR ← X
AC ← 1
AC ← AC + MBR
PC ← AC
```

Does JNS permit recursive calls?

No, PC is stored at address X, and we jump to address X+1. You can't do this repeatedly!

Clear Instruction

- `CLEAR` instruction
  - Set the contents of the accumulator to all zeroes.
- RTL for CLEAR: **AC ← 0**

MARIE Instructions (Full)

See table 4.7 in book!

Binary	Hex	Instruction	Meaning
0001	1	**LOAD X**	Load contents of address X into AC
0010	2	**STORE X**	Store contents of AC at address X
0011	3	**ADD X**	Add contents of address X to AC
0100	4	**SUBT X**	Subtract contents of address X from AC
0101	5	**INPUT**	Input value from keyboard into AC
0110	6	**OUTPUT**	Output value in AC to display
0111	7	**HALT**	Terminate program
1000	8	**SKIPCOND**	Skip next instruction on condition based on AC value
1001	9	**JUMP X**	Load value of X into PC
1010	10	**CLEAR**	Set AC to 0
1011	11	**ADDI X**	Add contents of address Mem[X] to AC
1100	12	**JUMPI X**	Load contents of address Mem[X] into PC
1101	13	**LOADI X**	Load contents of address Mem[X] into AC
1110	14	**STOREI X**	Store contents of AC at address Mem[X]