Term	Definition
MARIE	**M**achine **A**rchitecture that is **R**eally **I**ntuitive and **E**asy

4.8.1 The Architecture

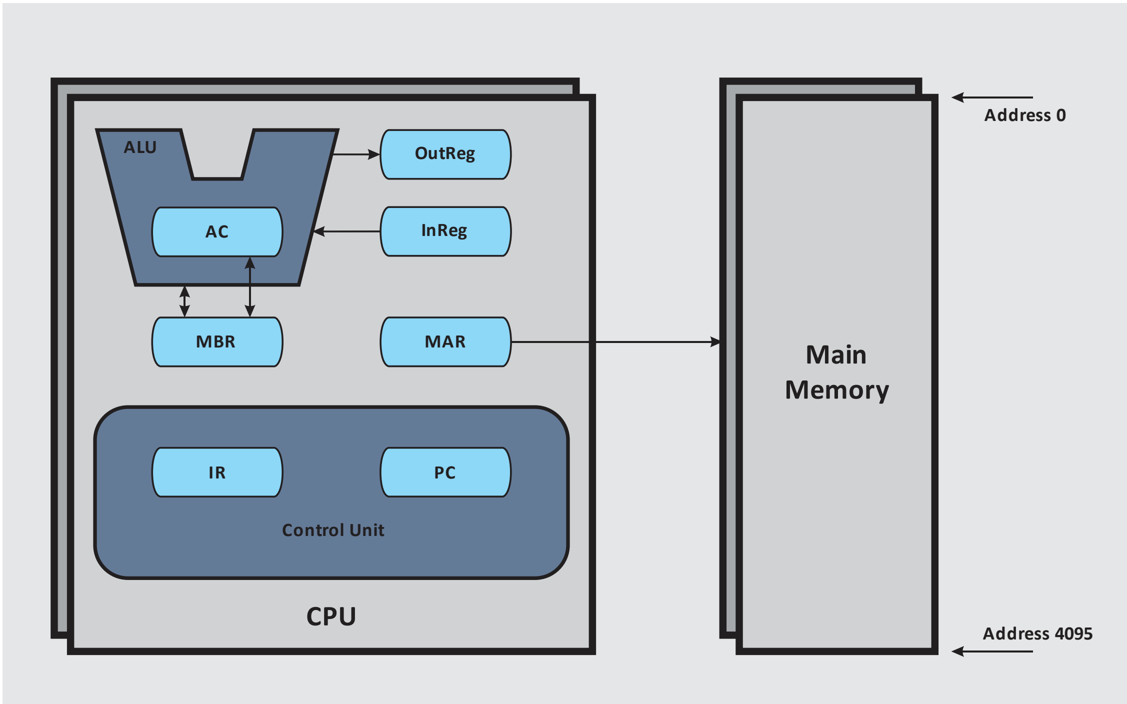

Figure 4.8 MARIE’s Architecture

Characteristics:

- 16-bit, binary, two’s complement
- Stored program, fixed word length
- Word (but not byte) addressable
- 4K (4096) words of main memory (this implies 12 bits per address)
- 16-bit data (words have 16 bits)
- 16-bit instructions: 4 bits for the opcode and 12 bits for the operand
- A 16-bit accumulator (AC)
- A 16-bit instruction register (IR)
- A 16-bit memory buffer register (MBR)
- A 12-bit memory address register (MAR)
- A 12-bit Program Counter (PC)
- An 8-bit input register
- An 8-bit output register

4.8.2 Registers and Buses

Register	Description
AC	The **accumulator** holds data values. This is a **general-purpose register** and it holds data that the CPU needs to process. Most computers today have multiple general-purpose registers.
MAR	The **memory address register** hold the memory address of the data being referenced.
MBR	The **memory buffer register** holds either the data just read from memory or the data ready to be written to memory.
PC	The **program counter** holds the address of the next instruction to be executed in the program.
IR	The **instruction register** holds the next instruction to be executed.
InREG	The **input register** holds data from the input device.
OutREG	The **output register** holds data for the output device.
status	The **status** or **flag register** holds information indicating various conditions, such as an overflow in the ALU, whether or not the result of an arithmetic or logical operation is zero, if a carry bit should be used in a computation, and when a result is negative.

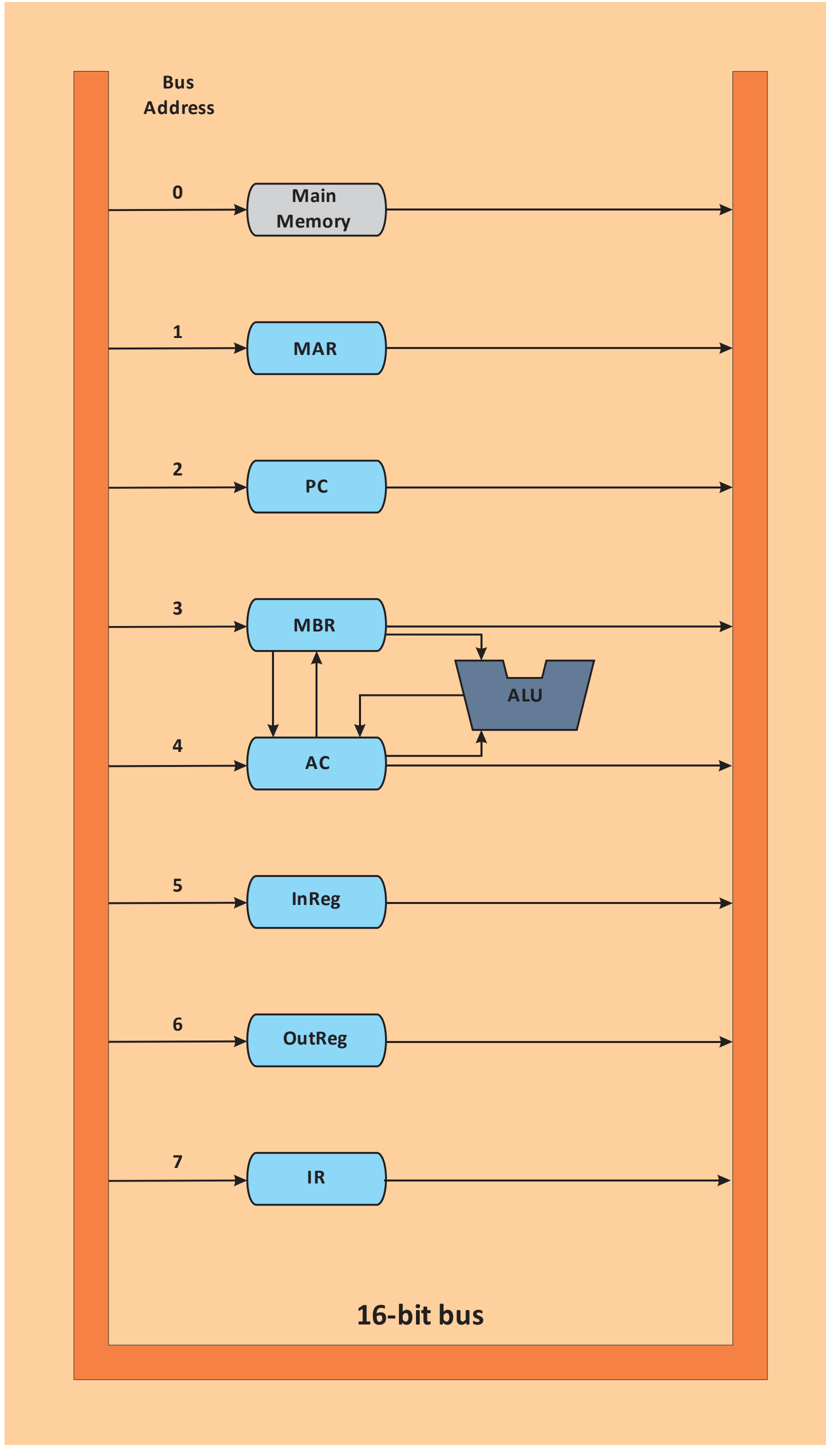

Figure 4.9 Datapath in MARIE

To copy data from one place to another, you must specify the origin and destination. For example, if you wish to copy data from the PC to the IR, then the origin is 2 (010_2) and the destination is 7 (111_2).

4.8.3 Instructions Set Architecture

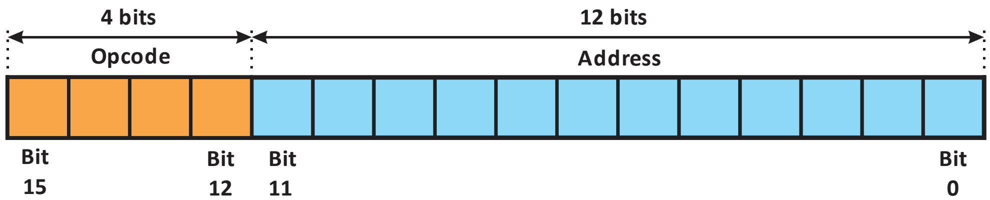

Figure 4.10 MARIE's Instruction Format

Instruction Number			
Bin	**Hex**	**Instruction**	**Meaning**
0000	0	JnS X	Store the PC address X and jump to X+1
0001	1	Load X	Load the contents of address X into the AC.
0010	2	Store X	Store the contents of the AC at address X.
0011	3	Add X	Add the contents of address X to the AC and store the result in the AC.
0100	4	Subt X	Subtract the contents of address X from the AC and store the result in the AC.
0101	5	Input	Input a value from the keyboard into the AC.
0110	6	Output	Output the value in the AC to the display.
0111	7	Halt	Terminate the program.
1000	8	Skipcond	Skip the next instruction on condition.
1001	9	Jump X	Load the value of X into the PC.
1010	A	Clear	Put all zeros in the AC
1011	B	AddI	Add indirect: Go to address X. Use the value at X as the actual address of the data operand to add to the AC.
1100	C	JumpI X	Jump indirect: Go to address X. Use the value at X as the actual address of the location to jump to.
1101	D	LoadI X	Load indirect: Go to address X. Use the value at X as the actual address of the operand to load into the AC.
1110	E	StoreI X	Store indirect: Go to address X. Use the value at X as the destination address for storing the value in the accumulator.
1111	F		

Table 4.2 MARIE's Instruction Set (Complete)

Conditional branching:

Bits 11 – 10	**Meaning**
00	"skip if the AC is negative"
01	"skip if the AC is equal to 0"
10	"skip if the AC is greater than 0"

	opcode				address (operand)											
					skip if the AC is negative											
	1	0	0	0	**0**	**0**										
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	opcode				address (operand)											
					skip if the AC is equal to 0											
	1	0	0	0	**0**	**1**										
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	opcode				address (operand)											
					skip if the AC is greater than 0											
	1	0	0	0	**1**	**0**										
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

4.8.4 Register Transfer Notation

Used to specify microoperations. A microoperation is an operation that can be performed in one clock cycle.

Notation	**Meaning**
$\leftarrow$	A transfer of information. For example, $MAR \leftarrow X$, the memory address register is assigned the value of X.
$M[X]$	M is memory and X is an address in the memory. $M[X]$ is value in memory and address X.

Instruction	**RTN**	**Explanation**
Load *X*	MAR ← X	Store the address portion of the instruction – X – 12 bits – into the memory address register.
	MBR ← M[MAR]	Store the contents of memory at the address in the memory address register to the memory buffer register.
	AC ← MBR	Store the value in the memory buffer register to the accumulator.

Instruction	RTN	Explanation
Store *X*	MAR ← X, MBR ← AC	Simultaneously, store the address portion of the instruction – X – 12 bits – into the memory address register *and* store the value of the accumulator in the memory buffer register.
	M[MAR] ← MBR	Store the value in the memory buffer register in memory at the address in the memory address register.
Add *X*	MAR ← X	Store the address portion of the instruction – X – 12 bits – into the memory address register.
	MBR ← M[MAR]	Store the value in the memory buffer register in memory at the address in the memory address register.
	AC ← AC + MBR	Add the value in the accumulator and the memory buffer register and store the sum in the accumulator.
Subt *X*	MAR ← X	Store the address portion of the instruction – X – 12 bits – into the memory address register.
	MBR ← M[MAR]	Store the value in the memory buffer register in memory at the address in the memory address register.
	AC ← AC-MBR	Subtract the value in the memory buffer register from the accumulator and store the difference in the accumulator.
Input	AC ← InREG	Store the value in the input register to the accumulator.
Output	OutREG ← AC	Store the value in the accumulator to the output register.
Halt		No operations are performed on registers; the machine simply ceases execution of the program.

Skipcond	**If** IR[11-10]=00 **then**
	    If AC<0 **then** PC←PC+1
	else If IR[11-10]=01 **then**
	    If AC=0 **then** PC←PC+1
	else If IR[11-10]=10 **then**
	    If AC>0 **then** PC←PC+1
		If bits 10 and 11 of the IR are 00 **then** …
		
		If bits 10 and 11 of the IR are 01 **then** …
		
		If bits 10 and 11 of the IR are 10 **then** …

Jump *X*	PC ← X (PC←IR[11-0])	Store the least significant 12 bits of the instruction register into the program counter.

Instruction	RTN	Explanation
Clear		Clear the accumulator. Assign zero to the accumulator.
	AC ← 0	Assign zero to the accumulator.
AddI X		Add Indirect. Find the address of the operand in X. First, fetch the address stored at X. Call this address Y. Then, fetch the value at Y. Call this value Z. Add Z to the accumulator.
	MAR ← X	
	MBR ← M[MAR]	
	MAR ← MBR	
	MBR ← M[MAR]	
	AC ← AC + MBR	
JumpI X		Jump Indirect. Find the address of the operand in X. First, fetch the instruction address stored at X. Call this address Y. Transfer control to the instruction at Y. Assign Y to the PC.
	MAR ← X	
	MBR ← M[MAR]	
	PC ← MBR	
LoadI X		Load Indirect. Find the address of the operand in X. First, fetch the address stored at X. Call this address Y. Next, fetch the value stored at Y. Assign Y to the accumulator.
	MAR ← X	
	MBR ← M[MAR]	
	MAR ← MBR	
	MBR ← M[MAR]	
	AC ← MBR	
StoreI X		Store Indirect. Find the address of the destination address at location X. First, fetch the address stored at X. Call this address Y. Assign the value in the accumulator to the memory location at address Y.
	MAR ← X	
	MBR ← M[MAR]	
	MAR ← MBR	
	MBR ← AC	
	M[MAR[← AC	