Instruction	RTN	Explanation
JnS X		Store the PC at address X and jump to X+1
	MBR ← PC	
	MAR ← X	
	M[MAR] ← MBR	
	MBR ← X	
	AC ← 1	
	AC ← AC + MBR	
	PC ← AC	
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	

Instruction	RTN	Explanation
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] ← MBR	

Opcode	Instruction	RTN
0000	JnS *X*	MBR ← PC MAR ← X M[MAR] ← MBR MBR ← X AC ← 1 AC ← AC + MBR PC ← AC
0001	Load *X*	MAR ← X MBR ← M[MAR] AC ← MBR
0010	Store *X*	MAR ← X, MBR ← AC M[MAR] ← MBR
0011	Add *X*	MAR ← X MBR ← M[MAR] AC ← AC + MBR
0100	Subt *X*	MAR ← X MBR ← M[MAR] AC ← AC-MBR
0101	Input	AC ← InREG
0110	Output	OutREG ← AC
0111	Halt	

1000	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

1001	Jump *X*	PC←IR[11-0]
1010	Clear	AC ← 0

Table 4.7 MARIE's Full Instruction Set

Opcode	Instruction	RTN
1011	AddI *X*	MAR ← X MBR ← M[MAR] MAR ← MBR MBR ← M[MAR] AC ← AC + MBR
1100	JumpI *X*	MAR ← X MBR ← M[MAR] PC ← MBR
1101	LoadI *X*	MAR ← X MBR ← M[MAR] MAR ← MBR MBR ← M[MAR] AC ← MBR
1110	StoreI *X*	MAR ← X MBR ← M[MAR] MAR ← MBR MBR ← AC M[MAR] ← MBR

Table 4.7 MARIE's Full Instruction Set (Continued)

if *X*=*Y* **then**
    X:=*X****2**
else
    Y:=*Y*-*X*;

Example 4.3 if-then-else construction
High Level Language Representation

Hex Address	Symbol	Instruction		Comment
ORG	100			
100	If,	Load	X	Load the first value, X (12)
101		Subt	Y	Subtract the value of Y from X AC=12-20=-8
102		Skipcond	400	If the AC=0, skip the next instruction
103		Jump	Else	Jump to the Else-part if AC<>0
104	Then,	Load	X	Reload X so it can be doubled
105		Add	X	Double X
106		Store	X	Store the new value
107		Jump	Endif	Skip over the Else-part
108	Else,	Load	Y	Else-part, Load Y
109		Subt	X	Subtract X from Y 20-12=8
10A		Store	Y	Store Y-X in Y
10B	Endif,	Halt		Stop
10C	X,	Dec	12	
10D	Y,	Dec	20	

Example 4.3 if-then-else construction
MARIE Instruction Representation

```
char* s="Hello world";
while (s) {
    cout << *s;
    s++;
}
```

Example 4.4 Print a string using indirect addressing
High Level Language Representation

Hex Address	Symbol	Instruction		Comment
ORG	100			
100	Getch,	LoadI	ChPtr	
101		Skipcond	400	
102		Jump	Outp	
103		Halt		
104	Outp,	Output		
105		Load	ChPtr	
106		Add	One	
107		Store	ChPtr	
108		Jump	Getch	
109	One,	Hex	0001	
10A	ChPtr,	Hex	10B	
10B	String,	Dec	072	/H
10C		Dec	101	/e
10D		Dec	108	/l
10E		Dec	108	/l
10F		Dec	111	/o
110		Dec	032	/[space]
111		Dec	119	/w
112		Dec	111	/o
113		Dec	114	/r
114		Dec	108	/l
115		Dec	100	/d
116		Dec	033	/!
117		Dec	000	/[null]

Example 4.4 Print a string using indirect addressing
MARIE Instruction Representation

```
int X=20;
int Y=48;
double(X);
double (Y);
void double(int* Temp) {Temp=Temp+Temp;}
```

Example 4.5 function call
High Level Language Representation

Hex Address	Symbol	Instruction		Comment
ORG	100			
100		Load	X	Load the first value to be doubled
101		Store	Temp	Copy the value to parameter Temp
102		JnS	Subr	Call the function. Store the return address at Subr
103		Store	X	The doubled value is in the accumulator. Store the doubled value back to X
104		Load	Y	Load the second value to be doubled
105		Store	Temp	Copy the value to parameter Temp
106		JnS	Subr	Call the function. Store the return address at Subr
107		Store	Y	The doubled value is in the accumulator. Store the doubled value back to Y
108		Halt		Stop
109	X,	Dec	20	
10A	Y,	Dec	48	
10B	Subr,	Hex	0	Store the return address here.
10C		Load	Temp	
10D		Add	Temp	
10E		JumpI	Subr	Return through the return address stored at Subr
10F	Temp,	Dec	0	Parameter for Subr

Example 4.5 function call
MARIE Instruction Representation