Computer System Architechture

Probir Mondal

Types of instructions

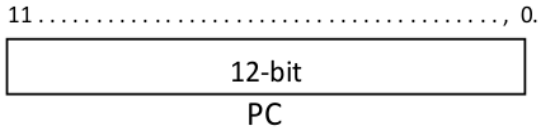
a) **Mnemonic Instructions or Machine Instructions**

b) **Pseudo Instructions**

a) Mnemonic Instructions or Machine Instructions:

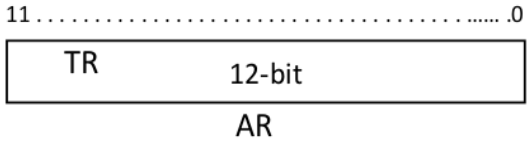
1. CPU Registers: 7 Registers

i) Program Counter(12-bit)



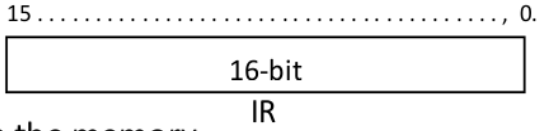
It holds the address of next instruction to be executed.

ii) Address Register(12-bit)



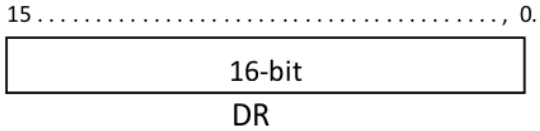
It holds the address of instruction/memory operand. It is directly connected to the Memory Unit.

iii) Instruction Register(16-bit)



It holds the instruction read from the memory.

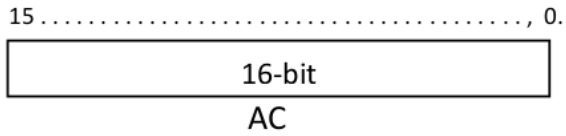
iv) Data Register(16-bit)



Operand read from memory is placed in data register.

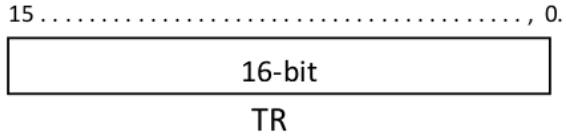
Contd...

v) Accumulator(16-bit)



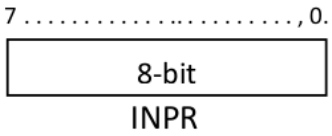
It holds the first operand before any arithmetic or logical operation and it also holds the result after the arithmetic or logical operation.

vi) Temporary Register(16-bit)



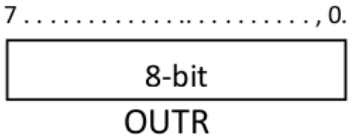
It holds temporary data during instruction execution.

vii) Input Register(8-bit)



It holds 8-bit input character from an Input Device.

viii) Output Register(8-bit)



It holds 8-bit character for an Output Device.

All registers are Special Purpose Registers (SPR).

Contd...

2. 1-bit Flip-Flop Used.

I is an 1-bit Flip-Flop which contains the left most bit position of the instruction. In case MRI, I is used to specify addressing mode (I=0/1) and otherwise it specifies RRI (I=0) or IOI(I=1).

E is an 1-bit Flip-Flop (Extended Accumulator Bit) which contains the output carry after the execution of ADD instruction. It can be set or complemented and it also contains the shifted bit after the execution of CIR or CIL.

S is an 1-bit Flip-Flop (Start-Stop Bit) which is initialized to 1. It specifies whether the CPU is Halt or Not. When CPU is halt, it set to 0, otherwise it is 1.

FGI (Control Flip-Flop) is a 1-bit Input Flag. It is initially set to 0. It is used for checking whether new information is available in the input device, when it is so, it is set 1 and new information is shifted to INPR and after that, information from INPR is transferred to AC and FGI is set to 0.

FGO (Control Flip-Flop) is a 1-bit Output Flag. It is initially set to 1. It is used for checking whether FGO flag is set or not, if it is set, then information from AC is shifted to OUTR and FGO is cleared to 0. The output device accepts the information and print it and FGO is set to 1.

R is an 1-bit Flip-Flop (Interrupt Flip-Flop) and it is initialized to 0. It is set to 1 when an interrupt is raised and it proceeds with an interrupt cycle in the next instruction cycle and, otherwise it proceeds with an Instruction cycle.

IEN is an 1-bit Flip-Flop (Interrupt Enable Flip-Flop) which is used to take decision whether or not to use interrupt facility(by ION or IOF).

3. Memory: It consists of 4096 (2^{12} memory locations) words; each of size is 16-bit.

INSTRUCTION FORMAT

Instruction Length – 16-bit. All instructions have the same length.

Instruction read from memory is placed in IR (Instruction Register).

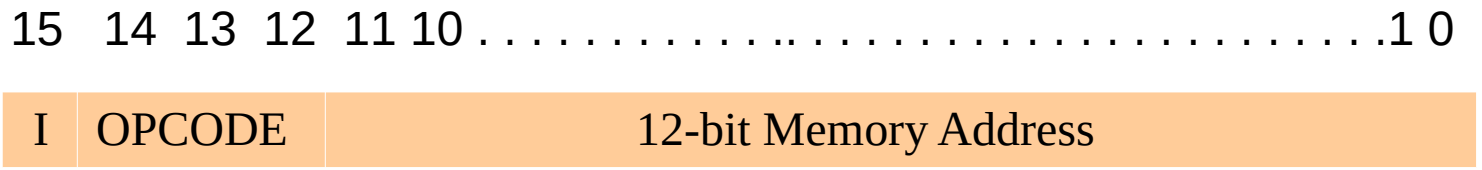
Instruction Types

i) Memory Reference Instructions(MRI).

ii) Register Reference Instructions(RRI).

Iii) Input-Output Instructions(IOI).

Memory Reference Instructions(MRI)



IR

It is specified by Operation Code ≠ 111 with a 0 or 1 in the left most bit position of the instruction.

a) Bit 15 of IR is transferred to 1-bit Flip-Flop I.

IR(15) – 1-bit specifies the Addressing Mode

i) IR(15)=0 for Direct Addressing Mode
[12-bit memory address contains the operand]

ii) IR(15)=1 for Indirect Addressing Mode
[12-bit memory address contains the address of the operand]

b) Next 3 bits IR (14,13,12) specifies Operation Code (Opcode).

- 000 (AND)
- 001 (ADD)
- 010 (LDA)
- 011 (STA)
- 100 (BUN)
- 101 (BSA)
- 110 (ISZ)

MRI	Symbol	Hex code	
		I=0(DA)	I=1(IAD)
1	AND	0XXX	8XXX
2	ADD	1XXX	9XXX
3	LDA	2XXX	AXXX
4	STA	3XXX	BXXX
5	BUN	4XXX	CXXX
6	BSA	5XXX	DXXX
7	ISZ	6XXX	EXXX

7 basic Instructions and total 14 instructions.

c) Next 12 bits IR (11, 10,……., 1,0) specifies operand address or address of location of the operand.

AND

The AND instruction implements the AND logic operation on the bit collection from the register and the memory word that is determined by the effective address. The result of this operation is moved back to the register.

AND XXX : AC <-- AC Λ M[XXX] 2)

ADD

The ADD instruction adds the content of the memory word that is denoted by the effective address to the value of the register.

ADD XXX : AC <-- AC + M[XXX]

LDA

The LDA instruction shares the memory word denoted by the effective address to the register.

LDA XXX : AC <-- M[XXX]

STA

STA saves the content of the register into the memory word that is defined by the effective address. The output is next used to the common bus and the data input is linked to the bus. It needed only one micro-operation.

STA XXX : M[XXX] <-- AC

BUN

The Branch Unconditionally (BUN) instruction can send the instruction that is determined by the effective address. They understand that the address of the next instruction to be performed is held by the PC and it should be incremented by one to receive the address of the next instruction in the sequence. If the control needs to implement multiple instructions that are not next in the sequence, it can execute the BUN instruction.

BUN XXX : PC <-- XXX

BSA

BSA stands for Branch and Save return Address. These instructions can branch a part of the program (known as subroutine or procedure). When this instruction is performed, BSA will store the address of the next instruction from the PC into a memory location that is determined by the effective address.

BSA XXX ; M[XXX] <-- PC, XXX <-- XXX + 1, PC <-- XXX

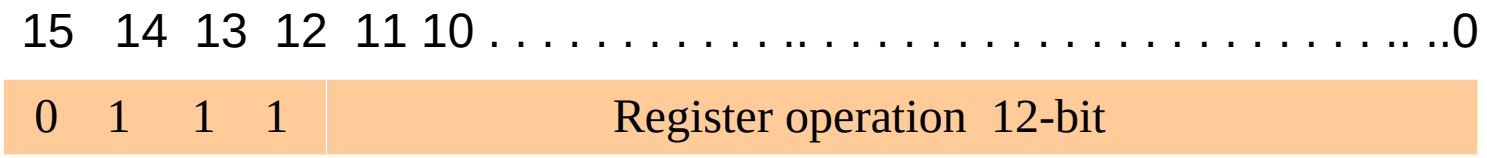
ISZ

The Increment if Zero (ISZ) instruction increments the word determined by effective address. If the incremented cost is zero, thus PC is incremented by 1. A negative value is saved in the memory word through the programmer. It can influence the zero value after getting incremented repeatedly. Thus, the PC is incremented and the next instruction is skipped.

ISZ XXX ; M[XXX] <-- M[XXX] + 1, IF (M[XXX] = 0) THEN PC <-- PC + 1

Maximum Number T-States=7 and Minimum Number T-States=5.
Maximum Number Machine Cycles=4 and Minimum Number Machine Cycles =1.

Register Reference Instructions (RRI)



IR

Register reference instruction is specified by Operation Code =111 with a **0** in the left most bit position of the instruction (IR (15) =0).

A register reference instruction specifies an operation on AC, S (Start-Stop bit) or E (Extended accumulator bit) or test on AC or E.

No operand from the memory is needed in this type of instruction. Each bit from position 11- bit to 0-bit specifies a unique operation. There are 12 instructions.

Instructions

RRI	Symbol	Hex code
1	CLA	7800
2	CLE	7400
3	CMA	7200
4	CME	7100
5	CIR	7080
6	CIL	7040
7	INC	7020
8	SPA	7010
9	SNA	7008
10	SZA	7004
11	SZE	7002
12	HLT	7001

Clear Instruction - i) CLA AC <- 0

ii)CLE E <- 0

Complement Instruction -

iii) CMA $AC \leftarrow \overline{AC}$

iv) CME $E \leftarrow \overline{E}$

Shift Instruction -v) CIR AC <-SHR(AC); AC(15) <- E; E <- AC(0)

vi) CIL AC <- SHL(AC); AC(0) <- E; E <- AC(15)

Increment Instruction -

vii) INC AC <- AC + 1

Skip and Test Instruction -

viii) SPA if (AC(15) = 0) then PC <-- PC+1

ix) SNA if (AC(15) = 1) then PC <-- PC+1

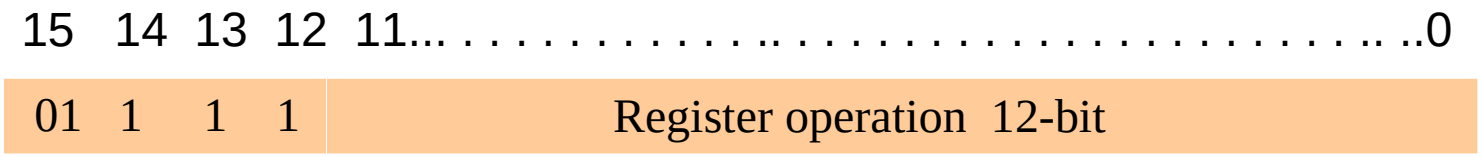
x) SZA if (AC= 0) then PC <-- PC+1

xi) SZE if (E = 0) then PC <-- PC+1

Halt Instruction - xii) Halt S <- 0

Maximum Number and Minimum Number T-States=4. Maximum Number and Minimum Number Machine Cycle =1.

Input-Output Instructions (IOI)



Input-Output instruction is specified by Operation Code =111 with a 1 in the left most bit position of the instruction (IR (15) =1).

An Input-Output Instruction specifies an operation on AC, FGI, FGO, INPR, OUTR or IEN (Interrupt Enable) or test on FGI or FGO.

No operand from the memory is needed in this type of instruction. Each bit from position 11-bit to 6-bit specifies a unique I/O operation. There are only 6 instructions.

Contd...

IOI	Symbol	Hex code
1	INP	F800
2	OUT	F400
3	SKI	F200
4	SKO	F100
5	ION	F080
6	IOF	F040

Input Character -i)INP;AC[7…..0] INPR; FGI 0

Output Character -ii)OUT;OUTR AC[7…..0]; FGO 0

Skip on Input Flag -iii)SKI;IF (FGI = 1) THEN PC PC+1;

Skip on Output Flag-iv)SKOIF (FGO = 1) THEN PC PC+1

Interrupt Enable On -v)IONIEN 1

Interrupt Enable Of -vi)IOF;IEN 0

Maximum Number and Minimum Number T-States=4. Maximum Number and Minimum Number Machine Cycle =1.

Pseudo Instructions(Assembler Directives)

i. **ORG N** : It specifies that hexadecimal number N is the memory address of the instruction/operand listed in the following line.

ii. **DEC N** : It converts signed decimal number N into binary and puts into a memory location.

iii. **HEX N** : It converts hexadecimal number N into binary and puts into a memory location either a hex constant or a label.

iv. **END** : It specifies the end of assembly language program(assembler does not consider any assembly language after end)

v. **CHR** : It puts a character into a memory location in lower byte using single quotes.

PROBLEM-1: Write an assembly language program to add n (n≥1) Integers (positive, negative and zero). The number of elements is stored at the location 150, numbers are stored after the location 150 and the sum is stored at the end.

Answer.

INSTRUCTION NO.	LABEL (Optional)	OPCODE/PSEUDO CODE (Mandatory)	ADDRESS (Optional)	INDIRECT ADDRESSING MODE (Optional)	COMMENTS (Optional)
1		ORG	100		PROGRAM STARTS AT LOC 100
2		LDA	APTR	I	
3		CMA			
4		INC			GET -N
5		STA	NPTR		
6		ISZ	APTR		
7		CLA			CLEAR AC
8		CLE			CLEAR E
9	BACK	ADD	APTR	I	
10		ISZ	APTR		
11		ISZ	NPTR		
12		BUN	BACK		
13		STA	APTR	I	
14		HLT			HALT CPU
15		ORG	120		
16	APTR	HEX	150		
17	NPTR	DEC	0		
18		ORG	150		
19		DEC	5		NUMBER OF ELEMENTS
20		DEC	7		
21		DEC	8		
22		DEC	9		
23		DEC	10		
24		DEC	12		
25		END			PROGRAM END

Problem: Write an assembly language program to find number of 1's in a word (16-bit). Word is stored at the memory location 030 and the number of 1's is stored after the memory location 030.

INSTRUCTION NO.	LABEL (Optional)	OPCODE/PSEUDO CODE (Mandatory)	ADDRESS (Optional)	INDIRECT ADDRESSING MODE (Optional)	COMMENTS (Optional)
1		ORG	100		PROGRAM STARTS AT LOC 100
2		CLA			AC <- 0
3		LDA	XXX		MEMORY WORD INTO AC
4	BACK	SZA			
5		BUN	NEXT		
6		BUN	DONE		
7	NEXT	ISZ	CNTR		INCREMENT COUNTER
8		STA	XXX		
9		LDA	ONE		
10		CMA			
11		INC			GET -1
12		ADD	XXX		A-1
13		AND	XXX		A & (A-1)
14		BUN	BACK		
15	DONE	HLT			HALT CPU
16		ORG	30		
17	XXX	HEX	FFFA		
18	CNTR	DEC	0		
19	ONE	DEC	1		
20		END			PROGRAM END

SYMBOL	ADDRESS
XXX	30
CNTR	31
ONE	32
DONE	1D
BACK	12

INPUT	OUTPUT
FFFA	14(0E)
0	0(0)
101	2(2)