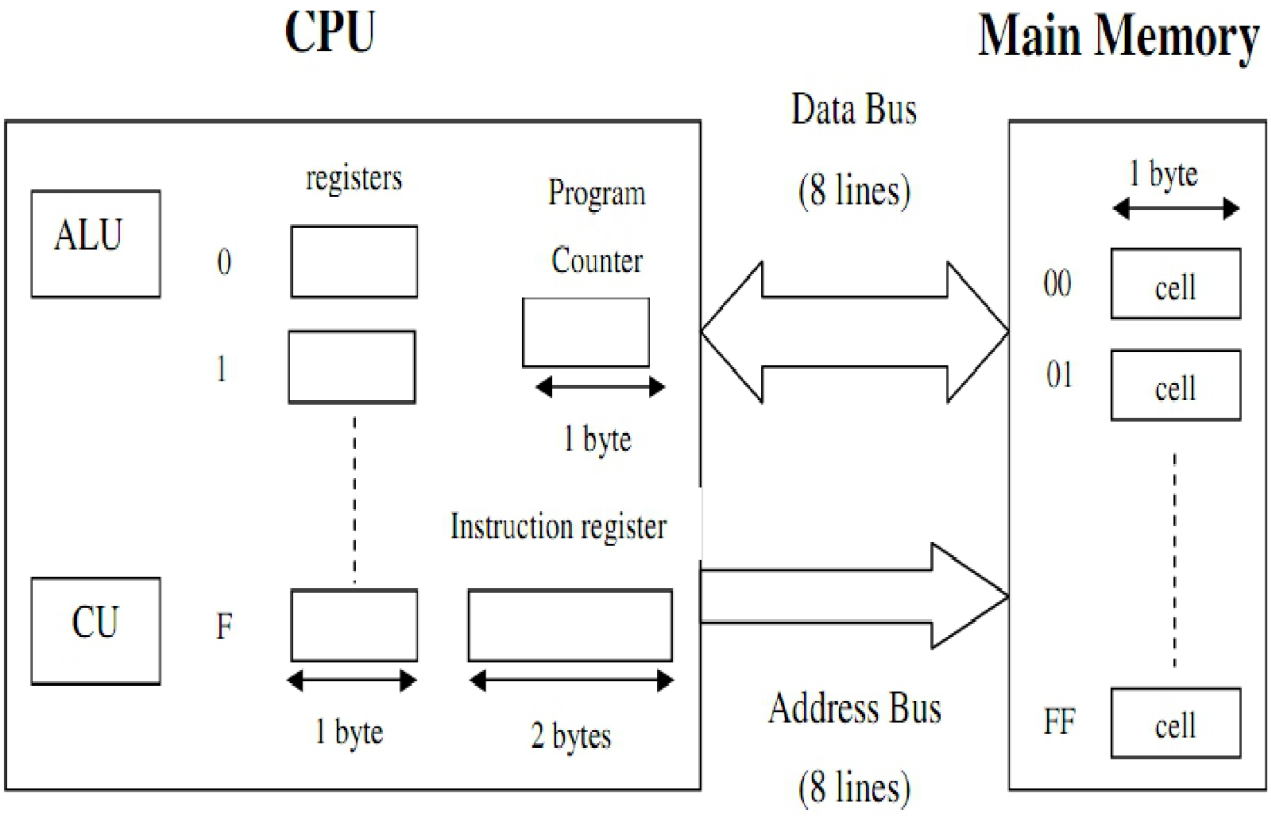
Hypothetical Algorithmic Machine Architecture



Machine Instructions: Example

Example (1): The following steps explain how a machine language program can be written to divide two values stored in memory:

step 1: load a register with a value from memory
step 2: load another register with the other value from memory
step 3: IF this second value is zero, jump to step 6
step 4: Divide the content of the first register by the second register and leave the result in a third register.
step 5: store the contents of the third register in memory.
step 6: stop

It is clear that:
steps: 1, 2, 5 → data transfer instructions
steps: 3, 6 → control instructions
step 4 → arithmetic instruction

Microprocessor: instruction set

- The microprocessor has a finite set of instructions (instruction set) that can be performed

- The instruction set include**:**
  1. Basic arithmetic and logic instructions
  2. Data transfer instructions
  3. Control instructions

Machine Instructions

The instructions set of a processor can be grouped into 3 categories as:

- "data transfer instructions":

  e.g. LOAD , STORE

  LOAD R0, AB → R0 ← AB

  LOAD R1 , M[AB] → R1 ← M[AB]

  STORE R5, F5 → M[F5] ← R5

- "arithmetic & logic instructions":

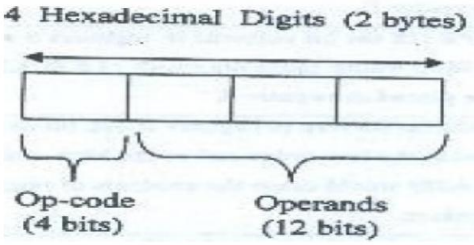
  ADD, AND, OR, XOR, SHIFT, ROTATE

- "control instructions"

  JUMP (or BRANCH), HALT (or STOP)

Machine Instructions as Bit Patterns

Both data and programs are stored in main memory in the same way as a stream of bits



- The general format of a machine instruction consists of two parts:

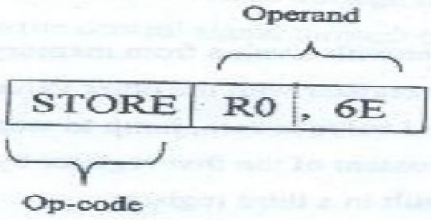
1- **The op-code:**

Which is the operation code of the instruction, such as: load, store, add.

2- **The operand**

Which provides more information about the operation specified by the op-code field.

Example (2): The format of the store instruction can be explained as follows:



Store the contents of register number 0 in memory cell of address 6E

The Instruction Set of the Hypothetical Algorithmic Machine

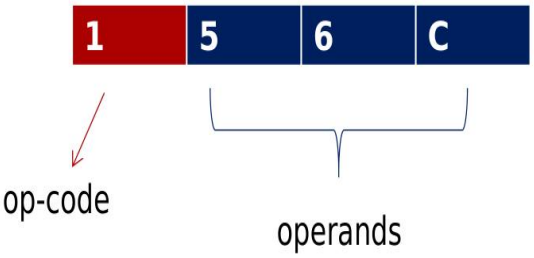
Op-code	Operand	Description
1	RXY	*LOAD* the register R with the bit pattern found in the memory cell whose address is XY. Example: 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	*LOAD* the register R with the bit pattern XY. Example: 20A3 would cause the value A3 to be placed in register 0.
3	RXY	*STORE* the bit pattern found in register R in the memory cell whose address is XY. Example: 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	0RS	*MOVE* the bit pattern found in register R to register S. Example: 40A4 would cause the contents of register A to be copied into register 4.
5	RST	*ADD* the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. Example: 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.
6	RST	*ADD* the bit patterns in registers S and T as though they represented values in floating-point notation and leave the floating-point result in register R. Example: 634E would cause the values in registers 4 and E to be added as floating-point values and the result to be placed in register 3.
7	RST	*OR* the bit patterns in registers S and T and place the result in register R. Example: 7CB4 would cause the result of ORing the contents of registers B and 4 to be placed in register C.
8	RST	*AND* the bit patterns in registers S and T and place the result in register R. Example: 8045 would cause the result of ANDing the contents of registers 4 and 5 to be placed in register 0.
9	RST	*EXCLUSIVE OR* the bit patterns in registers S and T and place the result in register R. Example: 95F3 would cause the result of EXCLUSIVE ORing the contents of registers F and 3 to be placed in register 5.

The Instruction Set of the Hypothetical Algorithmic Machine...

Op-code	Operand	Description
A	R0X	*ROTATE* the bit pattern in register R one bit to the right X times. Each time place the bit that started at the low-order end at the high-order end. Example: A403 would cause the contents of register 4 to be rotated 3 bits to the right in a circular fashion.
B	RXY	*JUMP* to the instruction located in the memory cell at address XY if the bit pattern in register R is equal to the bit pattern in register number 0. Otherwise, continue with the normal sequence of execution. Example: B43C would first compare the contents of register 4 with the contents of register 0. If the two were equal, the execution sequence would be altered so that the next instruction executed would be the one located at memory address 3C. Otherwise, program execution would continue in its normal sequence.
C	000	*HALT* execution. Example: C000 would cause program execution to stop.

Instruction: 156C

Op-code	Operand	Description
1	RXY	*LOAD* the register R with the bit pattern found in the memory cell whose address is XY. Example: 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.



Bit Pattern

Load register 5 with the contents of memory cell whose address 6C

Symbolic Form R5 ← M[6C]

Example(1) Use the instructions set table of the hypothetical CPU to translate the following machine language program:

step1 156C

step2 166D

step3 5056

step4 306E

step5 C000

Op-code	Operand	Description
1	RXY	*LOAD* the register R with the bit pattern found in the memory cell whose address is XY. Example: 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	*LOAD* the register R with the bit pattern XY. Example: 20A3 would cause the value A3 to be placed in register 0.
3	RXY	*STORE* the bit pattern found in register R in the memory cell whose address is XY. Example: 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	0RS	*MOVE* the bit pattern found in register R to register S. Example: 40A4 would cause the contents of register A to be copied into register 4.
5	RST	*ADD* the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. Example: 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.

Solution:

Step1: Load register 5 with the contents of memory cell whose address 6C

Step2: Load register 6 with the contents of memory cell whose address is 6D

Step3: ADD the contents of register 5 to that of register 6 , and put the result in register 0.

Step4: Store the contents of register 0 in the memory cell whose address is 6E.

Step5: Halt (i.e. stop).

Op-code	Operand	Description
1	RXY	*LOAD* the register R with the bit pattern found in the memory cell whose address is XY. Example: 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	*LOAD* the register R with the bit pattern XY. Example: 20A3 would cause the value A3 to be placed in register 0.
3	RXY	*STORE* the bit pattern found in register R in the memory cell whose address is XY. Example: 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	0RS	*MOVE* the bit pattern found in register R to register S. Example: 40A4 would cause the contents of register A to be copied into register 4.
5	RST	*ADD* the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. Example: 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.

Example(2) Rewrite the previous program in symbolic form

Solution:

```
LOAD  R5 , M[6C]
LOAD  R6 , M[6D]
ADD   R0 , R5, R6
STORE R0 , 6E
HALT
```

```
R5 ← M[6C]
R6 ← M[6D]
R0 ← R5 + R6
M[6E] ← R0
HALT
```


Op-code	Operand	Description
1	RXY	*LOAD* the register R with the bit pattern found in the memory cell whose address is XY. Example: 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	*LOAD* the register R with the bit pattern XY. Example: 20A3 would cause the value A3 to be placed in register 0.
3	RXY	*STORE* the bit pattern found in register R in the memory cell whose address is XY. Example: 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	0RS	*MOVE* the bit pattern found in register R to register S. Example: 40A4 would cause the contents of register A to be copied into register 4.
5	RST	*ADD* the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. Example: 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.

Example(2) Rewrite the previous program in symbolic form

Solution:

```
LOAD  R5 , M[6C]
LOAD  R6 , M[6D]
ADD   R0 , R5, R6
STORE R0 , 6E
HALT
```



```
R5 ← M[6C]
R6 ← M[6D]
R0 ← R5 + R6
M[6E] ← R0
HALT
```

1	RXY	*LOAD* the register R with the bit pattern found in the memory cell whose address is XY. Example: 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	*LOAD* the register R with the bit pattern XY. Example: 20A3 would cause the value A3 to be placed in register 0.
3	RXY	*STORE* the bit pattern found in register R in the memory cell whose address is XY. Example: 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	0RS	*MOVE* the bit pattern found in register R to register S. Example: 40A4 would cause the contents of register A to be copied into register 4.
5	RST	*ADD* the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. Example: 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.

in English:

i) 40F4 ii) BADE iii) 15AB iv) 25AB

Solution:

i) 40F4 → 0100 0000 1111 0100

copy the contents of register F to register 4

ii) BADE → 1011 1010 1101 1110

jump to the instruction located at memory address DE if the contents of register A equal that of register 0 .

iii) 15AB → 0001 0101 1010 1011

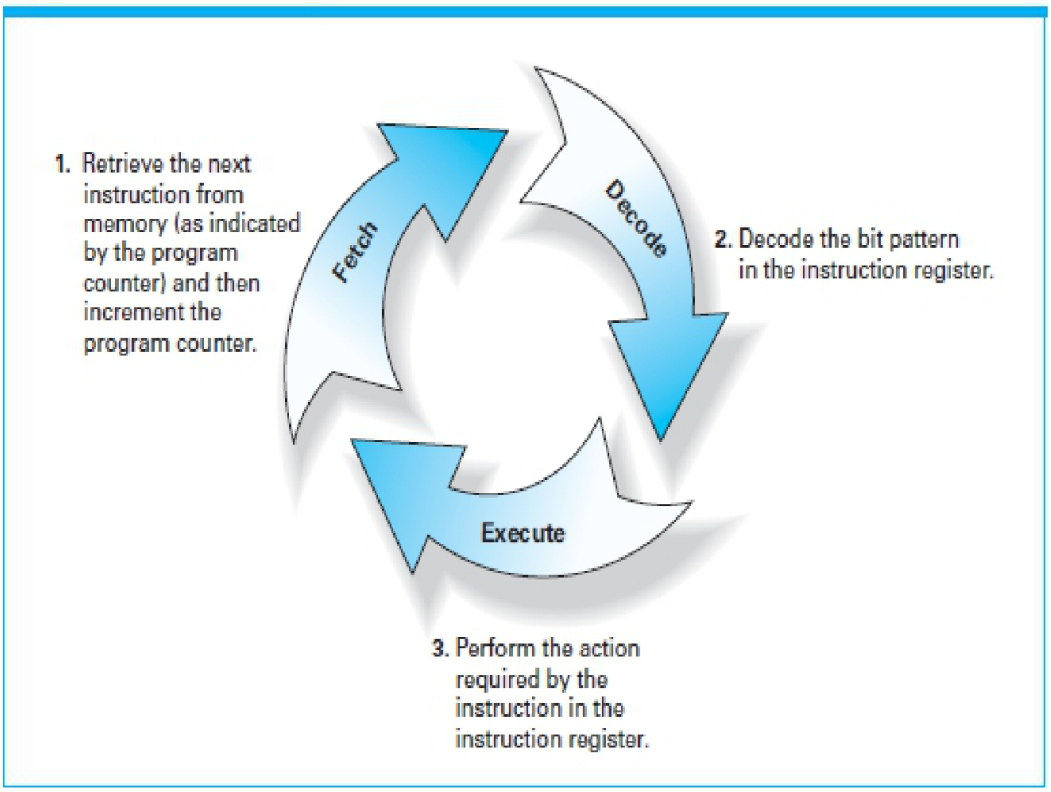
load register 5 with the contents of memory location AB

iv) 25AB → 0010 0101 1010 1011

load register 5 with value AB.

Instruction Execution Cycle

Figure 2.8 The machine cycle



Program Execution

Responsibility of

1. Operatng System (OS)

PC register: A1

2. CU: Machine Cycle
1- Fetching

IR register: 5056

PC register: A2

3. CU: Machine Cycle
2- Decoding

5 0 5 6

4. CU: Machine Cycle
3- Executing through calling appropriate ALU function

ALU: **Arithmatic Unit** is activated

A1	5056
A2	1526
A3	2061
A4	3504
A5	C000

My Program

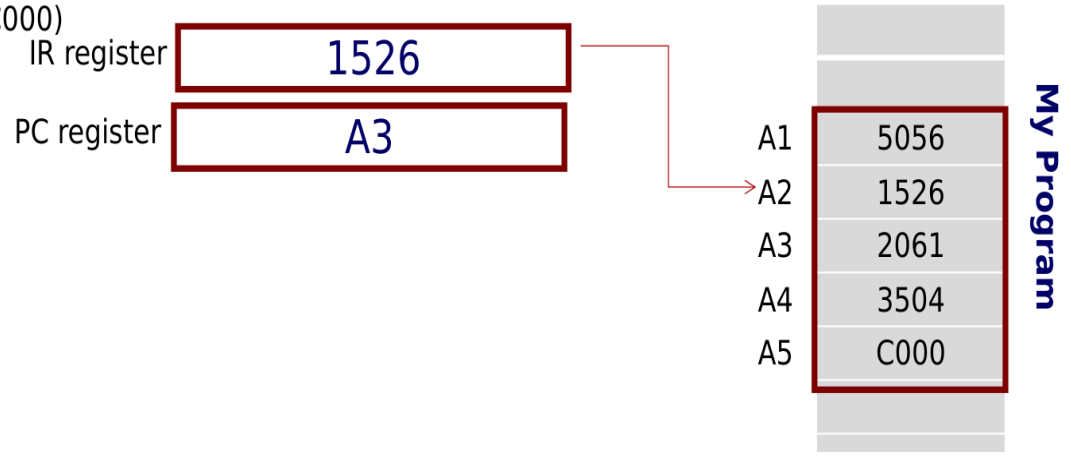
Responsibility of

5 ALU: Arithmatic Unit

R0 ← R5+R6

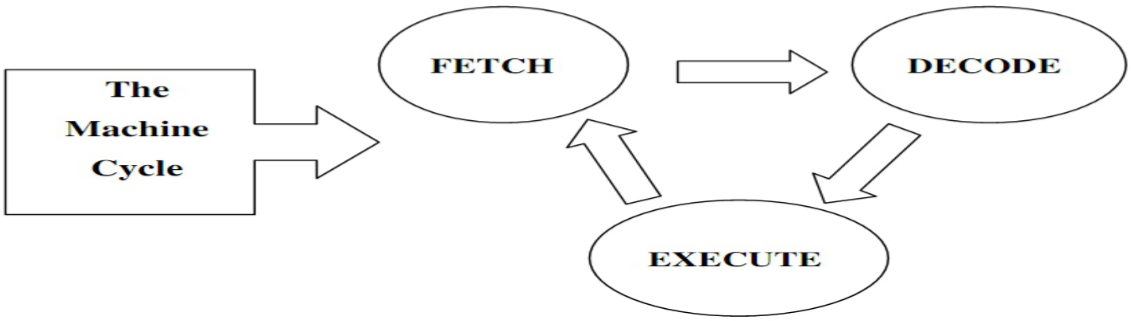
R5	0001 0111
+ R6	1110 0101
R0	1111 1100

6 CU: Repeat step 2 Untill Halt (C000)



Program Execution...

- To execute a program stored in main memory, the control unit of the CPU uses the special-purpose registers to do the job.
- The first step is that the start address of the program, such as 156C in the main memory should be placed in the "**program counter" register** by the operating system.
- Once the address of the first instruction to be executed is put in the "**program counter" register**, the control unit starts a "**repeating algorithm**" called **"machine cycle"**.



Program Execution: Machine Cycle

The machine cycle consists of 3 steps:

1. **Fetch**: which is fetching the instruction from the memory cell whose address is in the program counter, and putting it in the **instruction register**. At the end of fetch step, the **program counter register** is incremented to the next program instruction.

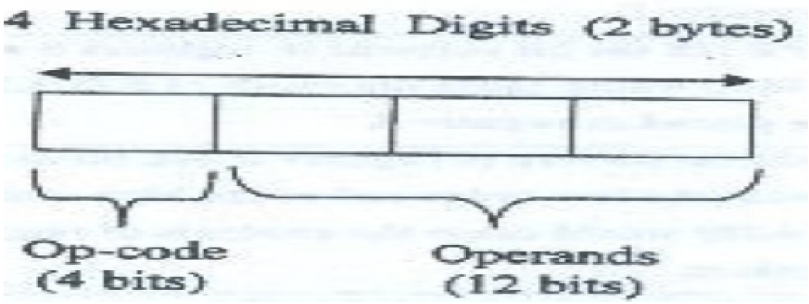
2. **Decode**: which is decoding the instruction contained in the **instruction register** and analyzing its op-code field , and operand field.

3. **Execute**: after decoding the instruction, the control unit activates the required circuity to perform the needed operation.

The fetch-decode-execute cycle continues until the "HALT" instruction is decoded in the **instruction register**, then before completing its execution the machine is informed that the program is completed and the machine cycle stops.

Since the instruction in the hypothetical CPU is 16 bits, and the size of each memory cell is 8 bits, then **each instruction of the program occupies two consecutive memory cells**.

- The format of the machine instruction is as follows:



Program Execution: Example

Op-code	Operand	Description
1	RXY	*LOAD* the register R with the bit pattern found in the memory cell whose address is XY. Example: 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	*LOAD* the register R with the bit pattern XY. Example: 20A3 would cause the value A3 to be placed in register 0.
3	RXY	*STORE* the bit pattern found in register R in the memory cell whose address is XY. Example: 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	0RS	*MOVE* the bit pattern found in register R to register S. Example: 40A4 would cause the contents of register A to be copied into register 4.
5	RST	*ADD* the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. Example: 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.

Example (4) This detailed example illustrates how the program given in example (1) be executed through the machine cycle of the control unit.

Let us assume that the program is stored in main memory as shown in the following table:

Address of memory cell	Contents of memory cell
A0	15
A1	6C
A2	16
A3	6D
A4	50
A5	56
A6	30
A7	6E
A8	C0
A9	00

Program Execution: Example: Instruction1

Thus, by the end of the first fetch phase, the special-purpose registers contain the following data: Program counter: A2

Instruction register: 156C

Next, the decode phase starts. The control unit analyzes the instruction in the instruction register and concludes that it is to load register 5 with the contents of memory cell at address 6C. This activity is executed during the execution phase. Then, the control unit returns to the fetch phase of the machine cycle.

Program Execution: Example: Instruction2

- During the second fetch phase, the control unit obtains the instruction 166D from the two memory cells of addresses A2 and A3. The control unit places the instruction in the instruction register and increments the program counter to A4. Thus, by the end of the second fetch phase

Thus, by the end of the second fetch phase, the special-purpose registers contain the following data: Program counter: A4 Instruction register: 166D

Program Execution: Example: Instruction3

- Now, the control unit decodes the instruction 166D and concludes that it is to load register 6 with the contents of memory location 6D. Then, it executes this operation.

- The control unit starts the third fetch phase and extracts the instruction at memory locations A4 and A5 which is 5056. Then, it puts that instruction in the instruction register and increments the program counter to A6.

Thus, by the end of the third fetch phase, the special-purpose registers contain the following data: Program counter: A6 Instruction register: 5056

Program Execution: Example: Instruction4

- Now, the control unit decodes the instruction 5056 and concludes that it is to add the contents of register 5 to that of register 6 and put the result in register 0. Then, it executes this operation.

- The control unit starts the fourth fetch phase and extracts the instruction at memory locations A6 and A7 which is 306E. Then, it puts that instruction in the instruction register and increments the program counter to A8.

Thus, by the end of the fourth fetch phase, the special-purpose registers contain the following data:	Program counter: A8 Instruction register: 306E

Program Execution: Example: Instruction5

- Now, the control unit decodes and executes this instruction which is storing the contents of register 0 in the memory location 6E.

- The control unit starts the fifth fetch phase starts at memory location A8. After extracting the instruction, which is C000, the program counter is incremented to AA.

Thus, by the end of the fifth fetch phase, the special-purpose registers contain the following data:	Program counter: AA Instruction register: C000

- Now, the control unit decodes the instruction C000, and concludes that it is HALT instruction. During the execution phase of the HALT instruction, the machine cycle stops and the program is completed.

Program Execution: Example2

Op-code	Operand	Description
1	RXY	*LOAD* the register R with the bit pattern found in the memory cell whose address is XY. Example: 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	*LOAD* the register R with the bit pattern XY. Example: 20A3 would cause the value A3 to be placed in register 0.
3	RXY	*STORE* the bit pattern found in register R in the memory cell whose address is XY. Example: 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	0RS	*MOVE* the bit pattern found in register R to register S. Example: 40A4 would cause the contents of register A to be copied into register 4.
5	RST	*ADD* the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. Example: 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.

Example (5) Assume the following program:

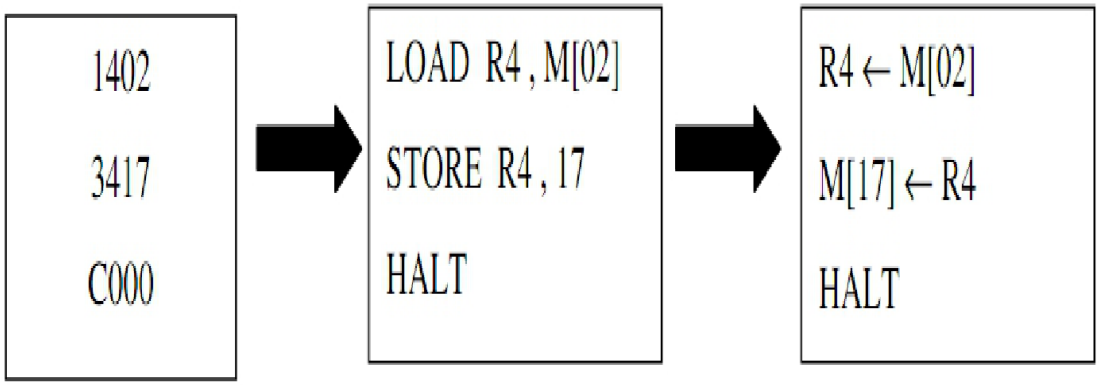
Address	Contents
00	14
01	02
02	34
03	17
04	C0
05	00

Starting the program from address 00, find the contents of memory cell whose address is 17 after the machine is halted.

Program Execution: Example2: Solution

Solution:

The program is:



Thus, the contents of memory location 17 is 34.

Program Execution: Example3

Op-code	Operand	Description
1	RXY	*LOAD* the register R with the bit pattern found in the memory cell whose address is XY. Example: 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	*LOAD* the register R with the bit pattern XY. Example: 20A3 would cause the value A3 to be placed in register 0.
3	RXY	*STORE* the bit pattern found in register R in the memory cell whose address is XY. Example: 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	0RS	*MOVE* the bit pattern found in register R to register S. Example: 40A4 would cause the contents of register A to be copied into register 4.
5	RST	*ADD* the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. Example: 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.

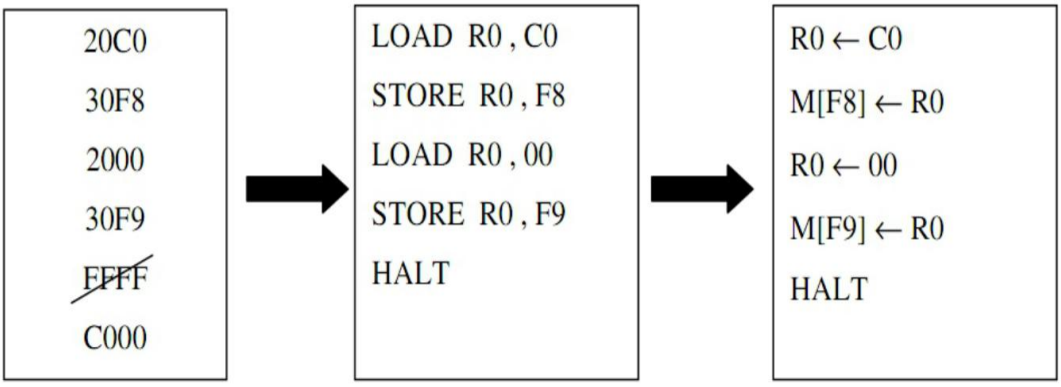
Example (6) Assume the following program:

Address	Contents
F0	20
F1	C0
F2	30
F3	F8
F4	20
F5	00
F6	30
F7	F9
F8	FF
F9	FF

Program Execution: Example3: Solution

If we start the machine with the program counter containing F0, what does the machine do when it reaches the instruction at address F8.

Solution: The program is:



The program halts at address F8 and this program is self-modifying because it modifies itself during execution.

Program Execution: Example4

Op-code	Operand	Description
1	RXY	*LOAD* the register R with the bit pattern found in the memory cell whose address is XY. Example: 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	*LOAD* the register R with the bit pattern XY. Example: 20A3 would cause the value A3 to be placed in register 0.
3	RXY	*STORE* the bit pattern found in register R in the memory cell whose address is XY. Example: 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	0RS	*MOVE* the bit pattern found in register R to register S. Example: 40A4 would cause the contents of register A to be copied into register 4.
5	RST	*ADD* the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. Example: 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.

Example (7) Suppose the memory cells at addresses A4 to B1 contains the following program:

Address	Contents
A4	20
A5	00
A6	21
A7	03
A8	22
A9	01
AA	B1
AB	B0

Address	Contents
AC	50
AD	02
AE	B0
AF	AA
B0	C0
B1	00

Program Execution: Example4...

Assuming program execution starts from address A4, answer the following questions:

a) What is the contents of register 0 after the first time the instruction at address AA is executed ?

b) What is the contents of register 0 after the second execution of the instruction at address AA ?

c) How many times the instruction at address AA is executed before the machine halts ?

Program Execution: Example4: Solution

Solution: The program is:

Starting address	Instructions	Meaning
A4	2000	R0 ← 00
A6	2103	R1 ← 03
A8	2201	R2 ← 01
AA	B1B0	Jump to B0 if R1 = R0
AC	5002	R0 ← R0 + R2
AE	B0AA	Jump to AA (if R0 = R0)
B0	C000	HALT

a) 00

b) 01

c) four times

Note:

- The jump instruction at address AA is conditional, while the jump instruction at address AE is unconditional.

- Before completing the executing phase of the jump instruction, the machine places the required address of the next instruction (according to the jump instruction) in the program counter.

Logical Operations

- The logic operations that can be performed by the ALU are: AND, OR and XOR.

AND:

0	0	1	1
AND 0	AND 1	AND 0	AND 1
0	0	0	1

OR:

0	0	1	1
OR 0	OR 1	OR 0	OR 1
0	1	1	1

XOR:

0	0	1	1
XOR 0	XOR 1	XOR 0	XOR 1
0	1	1	0

Logical Operations: Example (AND-OR)

Example (1):

01001011	10000011	11111111
AND 10101011	AND 11101100	AND 00101101
00001011	10000000	00101101

Example (2):

01001011	10000011	11111111
OR 10101011	OR 11101100	OR 00101101
11101011	11101111	11111111

Logical Operations: Example (XOR)

Example (3):

01001011	10000011	11111111
XOR	XOR	XOR
10101011	11101100	00101101
11100000	01101111	11010010

Note:
The above logical operations are also called "**logical masks**", or "**logical filters**"