GOAL: To develop machine's language program using MSTC processor's. The program should reserve memory for 3 integers (each four bytes long). The first integer location should be set to 100, second to 200 and the third one to the summation of first two.

Solution:

STAGE-I : PSEUDOCODE LEVEL.

Reserve M[1000:1003] for integer 1

Reserve M[1004:1007] for integer 2.

Reserve M[1008:1011] for integer 3.

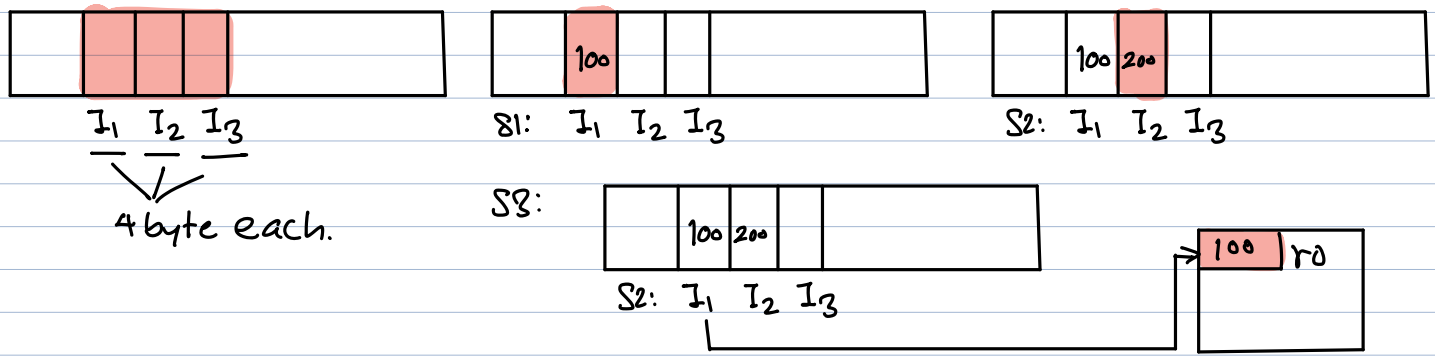
CPU-Step-1: M[1000:1003] ⟵ 100

CPU-Step-2: M[1004:1007] ⟵ 200

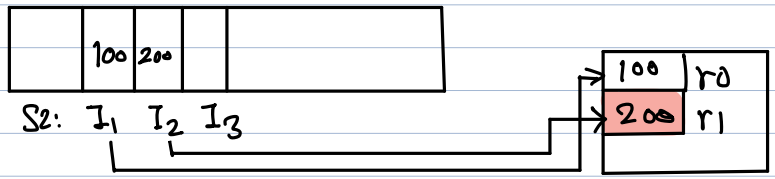
CPU-Step-3: R0 ⟵ M[1000:1003]

CPU-Step-4: R1 ⟵ M[1004:1007]

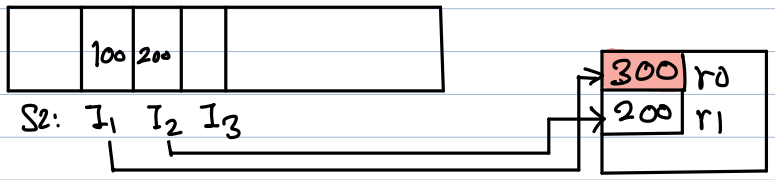
[Remember the discussion of the move instruction. You can recall that 4 byte wide constant can be moved into 4 byte memory and 4 byte memory can be moved into 4 byte register by using the move instruction. And therefore only one CPU step each is enough for first four operations].



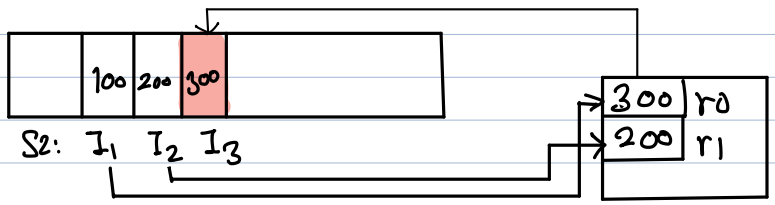
S4:



CPU-Step-5: $R_0 \leftarrow R_0 + R_1$



CPU-Step-6: $M[1008:1011] \leftarrow R_0$.



STAGE-II: MACHINE LANGUAGE PROGRAM but using decimal numbers.

Reserve M[1000:1003] for integer 1

Reserve M[1004:1007] for integer 2.

Reserve M[1008:1011] for integer 3

CPU-Step-I: $M[1000:1003] \leftarrow 100$

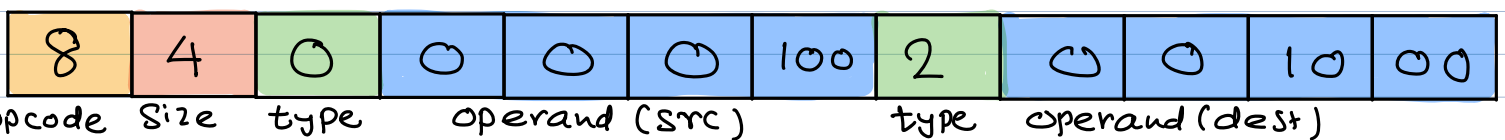
Instruction: mov, Opcode = 8.

Size of operands = 4.

Number of operands = 2.

1st operand = source operand: type = constant, number = 100

2nd operand = destination operand: type = memory, base addr = 1000



CPU-Step-II : M[1004:1007] ← 200

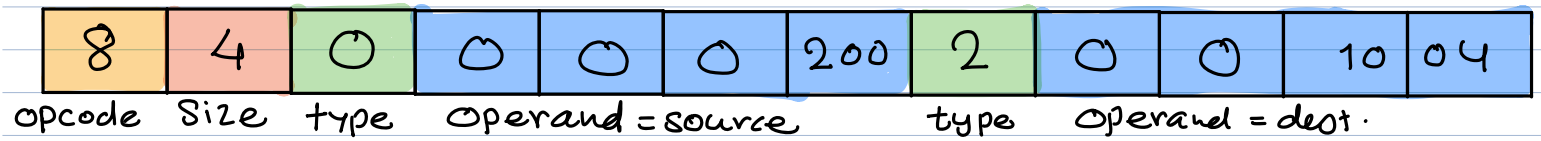
Instruction: mov. Opcode: 8

Size of operand = 4 bytes.

Number of operands = 2.

Operand 1 = source operand: type = constant number = 200

Operand 2 = destination operand, type = memory, base address = 1004,



CPU-Step-III R0 ← M[1000:1003]

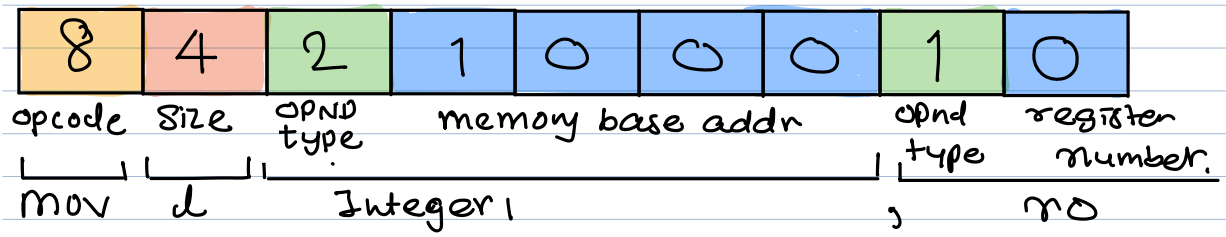
Instruction to be used: mov, opcode = 8

Size of the operand = 4 bytes.

Number of operands required by mov = 2.

Source operand: type = memory, base address = 1000

Destination operand: type = register, register number = 0.



Premature optimization is the root of evils.

— Donald Knuth!

CPU-Step-4: R1 ← M[1004:1007]

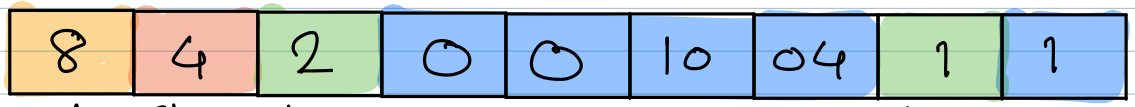
Instruction: mov, opcode = 8

Size of operands = 4.

Number of operands = 2.

Operand-1 = Source operand: type = memory base address = 1004.

Operand 2 = Destination operand : type = register, register number = 1.



CPU - Step - 5 :

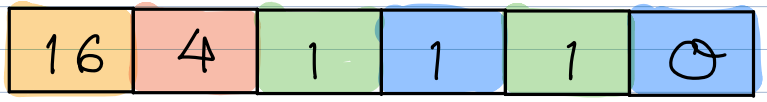
$R_0 \leftarrow R_0 + R_1$

Instruction: Add, Opcode = 16.

Operand size = 4, number of operand = 2.

1st operand source: type = reg, #number = 1

2nd operand : destination: type = reg # number = 0

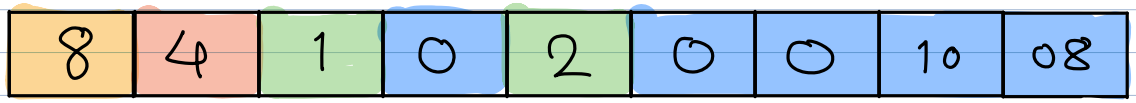


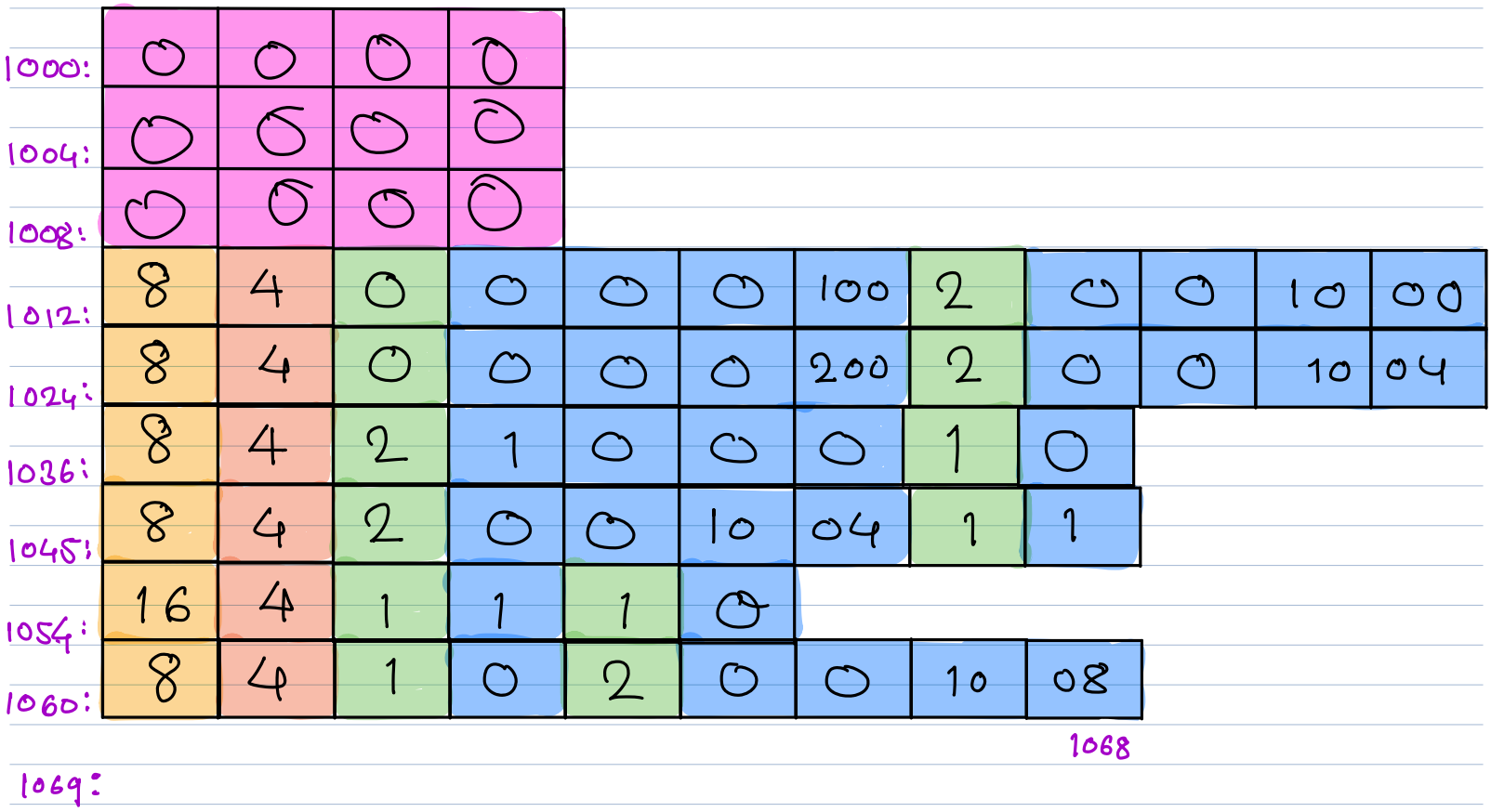
CPU - Step - 6 : M[1008 : 1011] $\leftarrow$ R_0.

Instruction = mov, opcode = 8, operand size = 4, # of operands = 2

Operand - 1 = source : type = register, number = 0

Operand - 2 : destination : type = memory base addr = 1008





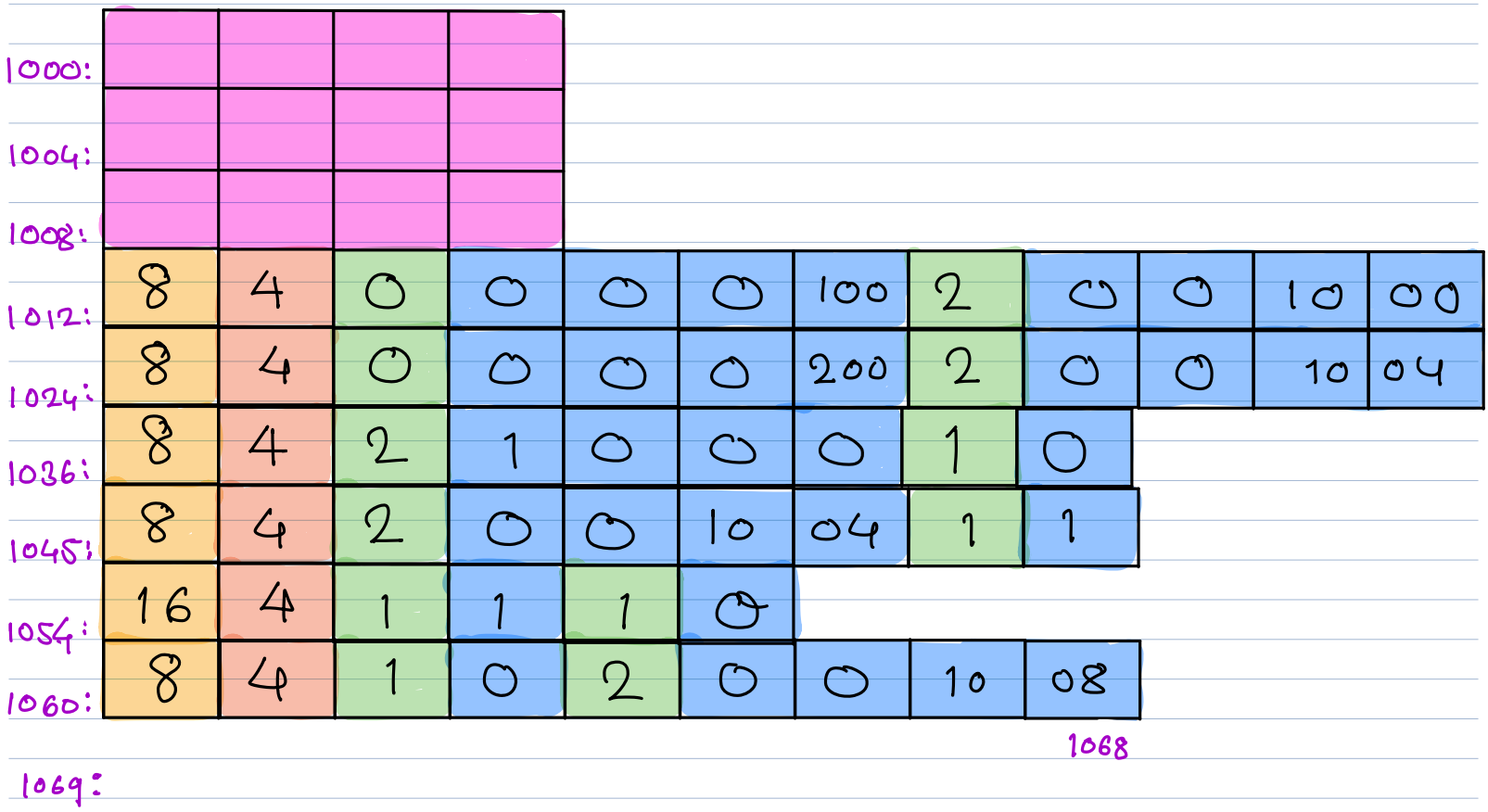
- section .bss
  - comm n1, 4, 4
  - comm n2, 4, 4.
  - comm sum, 4, 4.
  - section .text

main:

```
movl   $100, n1
movl   $200, n2
movl   n1, %eax
movl   n2, %edx
addl   %edx, %eax
movl   %eax, sum
```

```
int n1, n2, sum;
main()
{
    n1=100;
    n2=200;
    sum= n1+n2;
}
```

STAGE-III : Binary Machine Language Code.



$(8)_{10} = (1000)_2 = 0000\ 1000$

$(16)_{10} = (10000)_2 = 0001\ 0000$

$(4)_{10} = (100)_2 = 0000\ 0100$

$(2)_{10} = (10)_2 = 0000\ 0010$

$(1)_{10} = (1)_2 = 0000\ 0001$

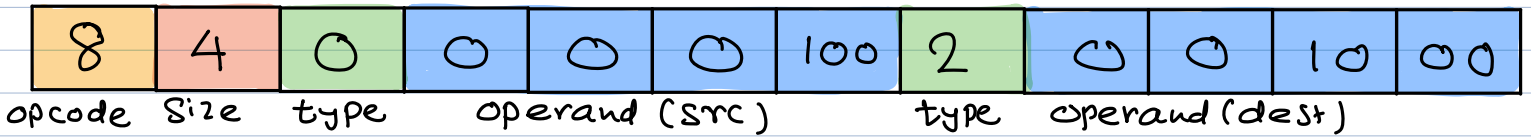
$(100)_{10} = (1100100)_2 = 0110\ 0100$

$(200)_{10} = (1100\ 1000)_2$ 00000000 00000000 00000000 11001000

$(1000)_{10} = (11\ 11101000) = 0000\ 0011\ \ 11101000$

$(1004)_{10} = (11\ 11101100) = 00000011\ \ 11101100$

$(1008)_{10} = (11\ 11110000)_2 = 00000011\ 11110000$



On paper LSB = Least addr.

On paper MSB = Most addr.

On paper

1101 0011 1100 0101 1111 0010 1001 0110

Most Significa

Least Signi

On paper ⟶ computer bytes.

3 1 ficat

On paper LSB in 1 yte with east addr.

On paper MSB in Byte with most addr.

10010110	11110010	11000101	11010011
5000	5001	5002	5003

11010011	11000101	11110010	10010110
5003	5002	5001	5000

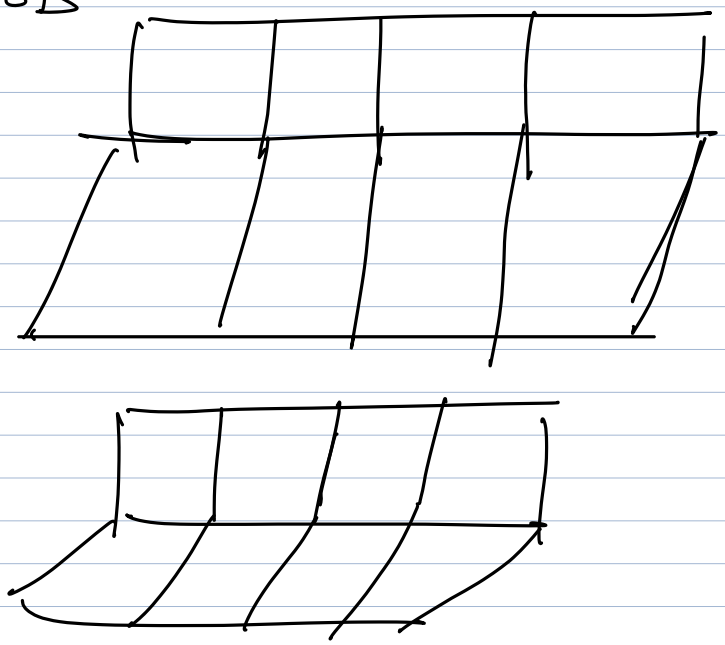
5000	10010110
5001	11110010
5002	11000101
5003	11010011

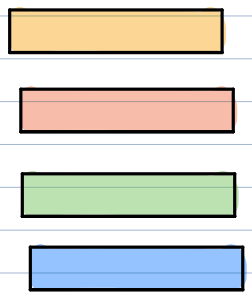
5003	11010011
5002	11000101
5001	11110010
5000	10010110

5000 10010110
5001 11110010
5002 11000101
5003 11010011

On paper

1101 0011 1100 0101 1111 0010 1001 0110
MSB LSB





8	4	0	0	0	0	100	2	0	0	10	00
opcode	Size	type	operand (src)				type	operand (dest)			

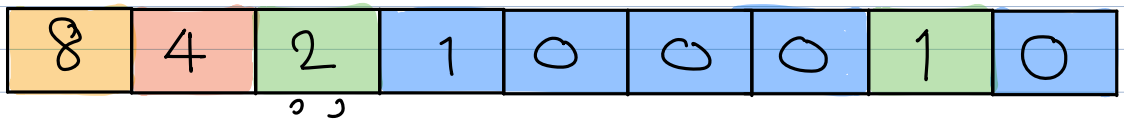
00001000	00000010	00000000	01100100	00000000	00000000	00000000
00000010	11101000	00000000	00000000	00000000		

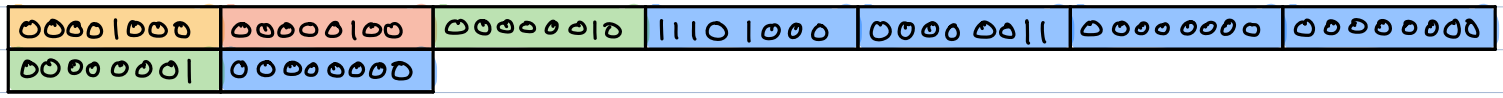
CPU Step-2

8	4	0	0	0	0	200	2	0	0	10	04
opcode	Size	type	operand = source				type	operand = dest.			

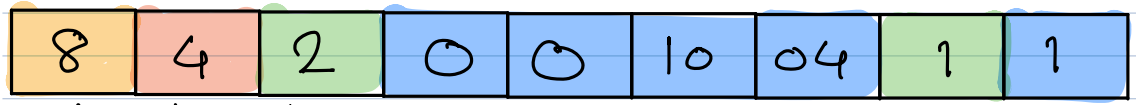
00001000	00000100	00000000	11001000	00000000	00000000	00000000
00000010	11101100	00000011	00000000	00000000		

CPU Step - 3

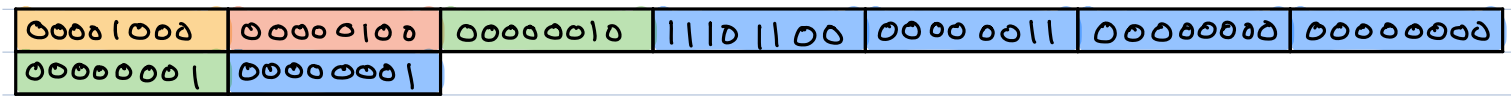




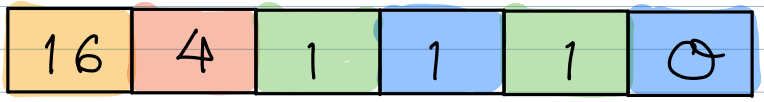
CPU - Step- 4.

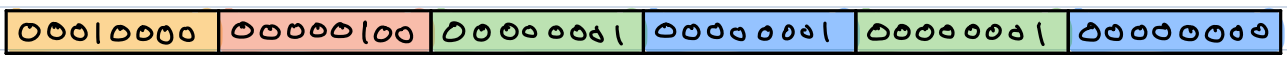


opcode size type operand 1 = source type operand 2 desi.

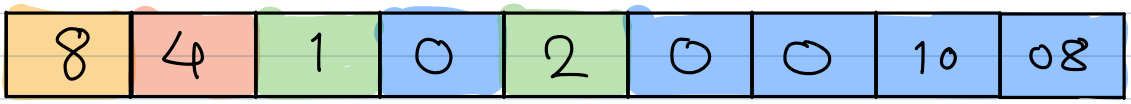


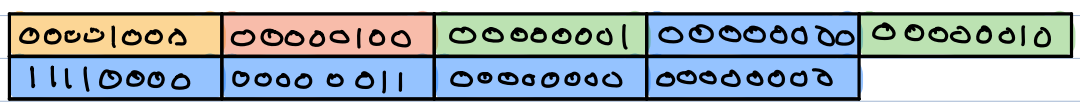
CPU - Step - 5

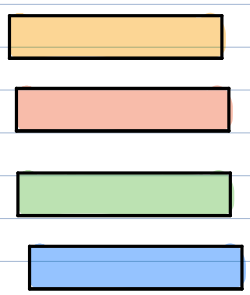




CPU - Step - 6.







MACHINE LANGUAGE PROGRAM - I.

1000							
1004							
1008							
1012							
1024							
1036							
1045							
1054							
1060							

1069