IOT AND CLOUD COMPUTING LAB_4

Name: Manikandan P
RegNo: 2019202030

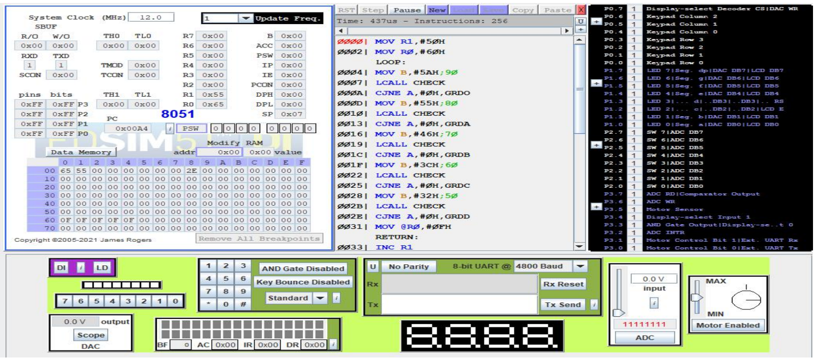
1. Write an ALP to grade(as per 2012 regulation) an array of student marks stored at memory location 5100H. The grade of each student must be stored in location 5200H. assume the last digit of memory address as the roll numbers of students.

CODE:

```
MOV R1,#50H
MOV R0,#60H
LOOP:
MOV B,#5AH;90
LCALL CHECK
CJNE A,#0H,GRDO
MOV B,#55H;80
LCALL CHECK
CJNE A,#0H,GRDA
MOV B,#46H;70
LCALL CHECK
CJNE A,#0H,GRDB
MOV B,#3CH;60
LCALL CHECK
CJNE A,#0H,GRDC
MOV B,#32H;50
LCALL CHECK
CJNE A,#0H,GRDD
MOV @R0,#0FH
RETURN:
INC R1
INC R0
CJNE R1,#55H,LOOP
SJMP ENDL
```

```
CHECK:
MOV A,@R1
DIV AB
RET
GRDO:
MOV @R0,#0H
SJMP RETURN
GRDA:
MOV @R0,#0AH
SJMP RETURN
GRDB:
MOV @R0,#0BH
SJMP RETURN
GRDC:
MOV @R0,#0CH
SJMP RETURN
GRDD:
MOV @R0,#0DH
SJMP RETURN
ENDL:
END
```

OUTPUT:



Time: 437us - Instructions: 256

```
0000| MOV R1,#50H
0002| MOV R0,#60H
      LOOP:
0004| MOV B,#5AH;90
0007| LCALL CHECK
000A| CJNE A,#0H,GRDO
000D| MOV B,#55H;80
0010| LCALL CHECK
0013| CJNE A,#0H,GRDA
0016| MOV B,#46H;70
0019| LCALL CHECK
001C| CJNE A,#0H,GRDB
001F| MOV B,#3CH;60
0022| LCALL CHECK
0025| CJNE A,#0H,GRDC
0028| MOV B,#32H;50
002B| LCALL CHECK
002E| CJNE A,#0H,GRDD
0031| MOV @R0,#0FH
      RETURN:
0033| INC R1
```

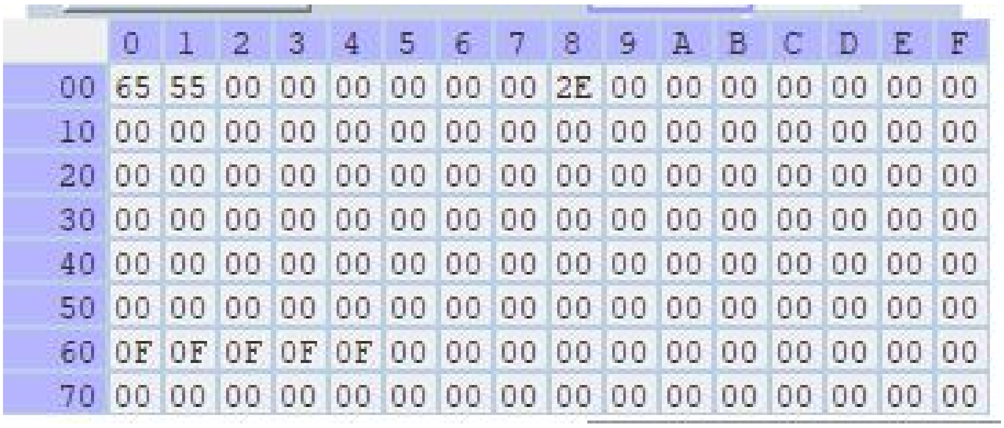
System Clock (MHz) 12.0

R7	0x00	B	0x00
R6	0x00	ACC	0x00
R5	0x00	PSW	0x00
R4	0x00	IP	0x00
R3	0x00	IE	0x00
R2	0x00	PCON	0x00
R1	0x55	DPH	0x00
R0	0x65	DPL	0x00
		SP	0x07

PC 0x00A4

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	65	55	00	00	00	00	00	00	2E	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	0F	0F	0F	0F	0F	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2021 James Rogers



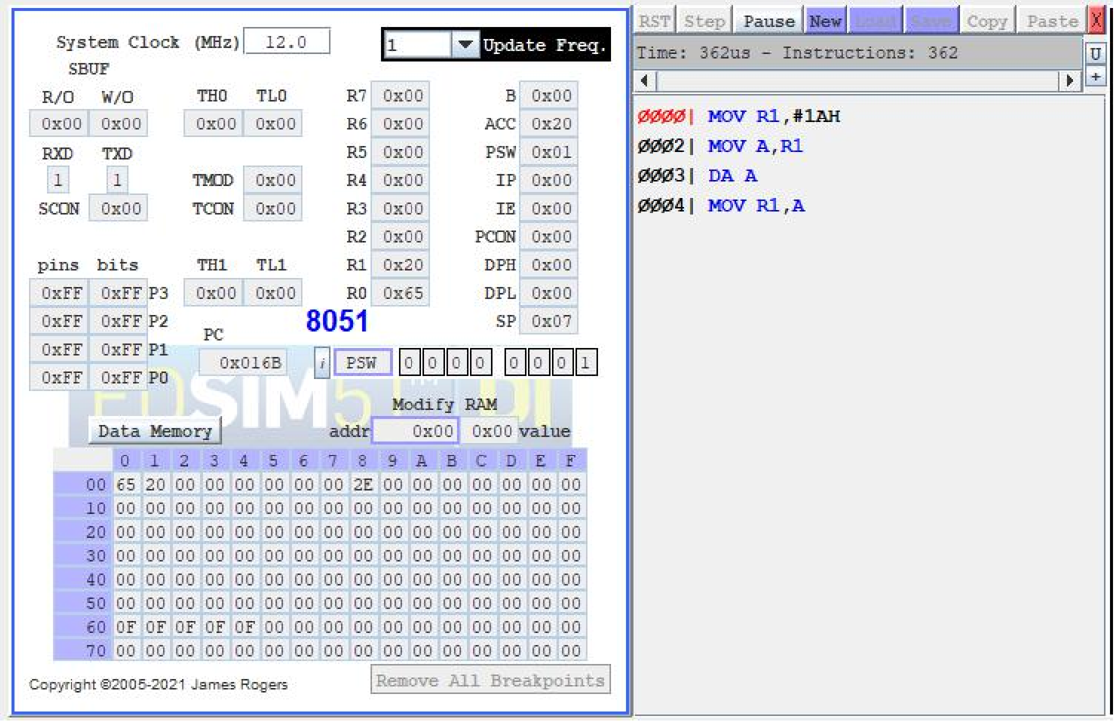
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	65	55	00	00	00	00	00	00	2E	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	0F	0F	0F	0F	0F	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

2. Write an ALP to convert hex to BCD:

CODE:

```
MOV R1,#1AH
MOV A,R1
DA A
MOV R1,A
```

OUTPUT:



System Clock (MHz) 12.0

Time: 362us - Instructions: 362

```
0000| MOV R1,#1AH
0002| MOV A,R1
0003| DA A
0004| MOV R1,A
```

SBUF R/O 0x00 W/O 0x00

RXD 1 TXD 1

SCON 0x00

TH0 0x00 TL0 0x00

TMOD 0x00 TCON 0x00

TH1 0x00 TL1 0x00

R7 0x00
R6 0x00
R5 0x00
R4 0x00
R3 0x00
R2 0x00
R1 0x20
R0 0x65

B 0x00
ACC 0x20
PSW 0x01
IP 0x00
IE 0x00
PCON 0x00
DPH 0x00
DPL 0x00
SP 0x07

pins bits
0xFF 0xFF P3
0xFF 0xFF P2
0xFF 0xFF P1
0xFF 0xFF P0

PC 0x016B

PSW 0 0 0 0 0 0 0 1

addr 0x00 0x00 value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	65	20	00	00	00	00	00	00	2E	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	0F	0F	0F	0F	0F	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2021 James Rogers

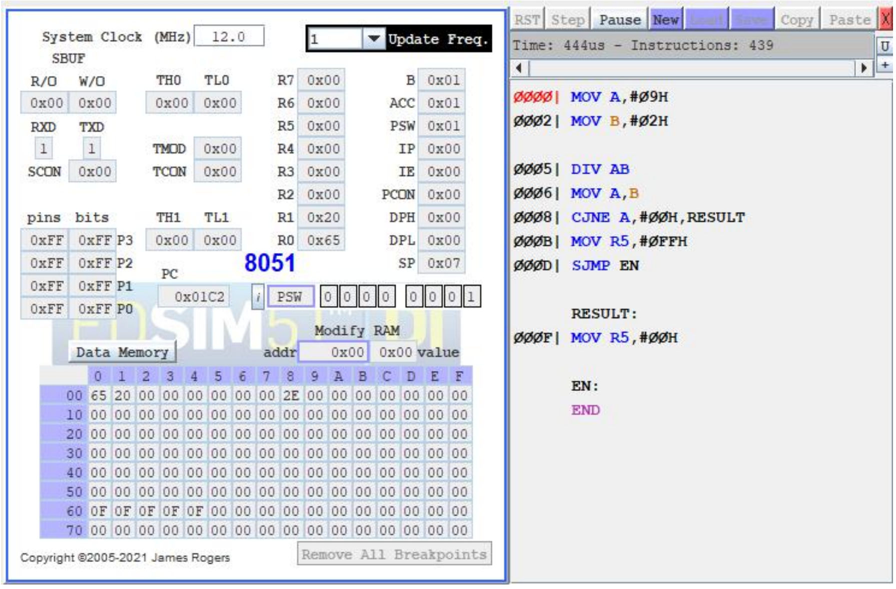
3. Write an ALP to check whether a given number is even or odd. If the number is even display FFh in R5 else display 00h

CODE:

```
MOV A,#09H
MOV B,#02H
DIV AB
MOV A,B
CJNE A,#00H,HELLO
MOV R5,#0FFH
SJMP EN
HELLO:
MOV R5,#00H
EN:
END
```

OUTPUT:

ODD NUMBER:



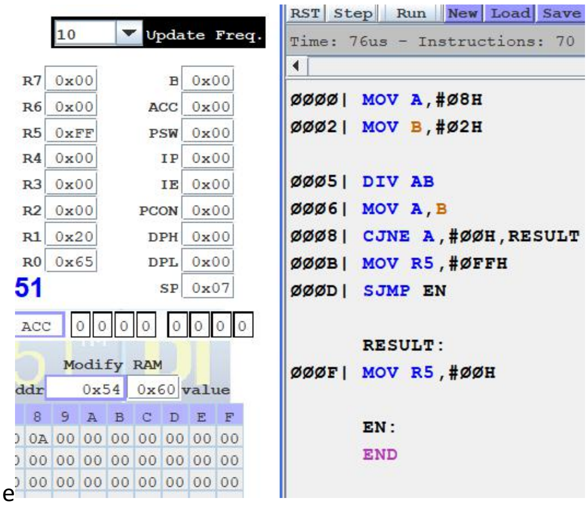
```
System Clock (MHz) 12.0      1  Update Freq.
SBUF
R/O   W/O     TH0   TL0    R7 0x00      B 0x01
0x00  0x00    0x00  0x00   R6 0x00    ACC 0x01
RXD   TXD                  R5 0x00    PSW 0x01
1     1       TMOD  0x00   R4 0x00     IP 0x00
SCON  0x00    TCON  0x00   R3 0x00     IE 0x00
                           R2 0x00   PCON 0x00
pins  bits    TH1   TL1    R1 0x20    DPH 0x00
0xFF  0xFF P3 0x00  0x00   R0 0x65    DPL 0x00
0xFF  0xFF P2         8051             SP 0x07
0xFF  0xFF P1  PC 0x01C2   PSW 0 0 0 0  0 0 0 1
0xFF  0xFF P0
                           Modify RAM
                      addr 0x00   0x00 value
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	65	20	00	00	00	00	00	00	2E	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	0F	0F	0F	0F	0F	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2021 James Rogers

```
Time: 444us - Instructions: 439

ØØØØ| MOV A,#Ø9H
ØØØ2| MOV B,#Ø2H

ØØØ5| DIV AB
ØØØ6| MOV A,B
ØØØ8| CJNE A,#ØØH,RESULT
ØØØB| MOV R5,#ØFFH
ØØØD| SJMP EN

      RESULT:
ØØØF| MOV R5,#ØØH

      EN:
      END
```

EVEN NUMBER



```
10  Update Freq.

R7  0x00      B     0x00
R6  0x00      ACC   0x00
R5  0xFF      PSW   0x00
R4  0x00      IP    0x00
R3  0x00      IE    0x00
R2  0x00      PCON  0x00
R1  0x20      DPH   0x00
R0  0x65      DPL   0x00
              SP    0x07

ACC  0 0 0 0  0 0 0 0

ddr 0x54  0x60 value
 8  9  A  B  C  D  E  F
0A 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

```
Time: 76us - Instructions: 70

ØØØØ| MOV A,#Ø8H
ØØØ2| MOV B,#Ø2H

ØØØ5| DIV AB
ØØØ6| MOV A,B
ØØØ8| CJNE A,#ØØH,RESULT
ØØØB| MOV R5,#ØFFH
ØØØD| SJMP EN

      RESULT:
ØØØF| MOV R5,#ØØH

      EN:
      END
```

e

4. Write an ALP to check whether a given number is prime or not. If the number is prime display FFh in R7 else display 00h

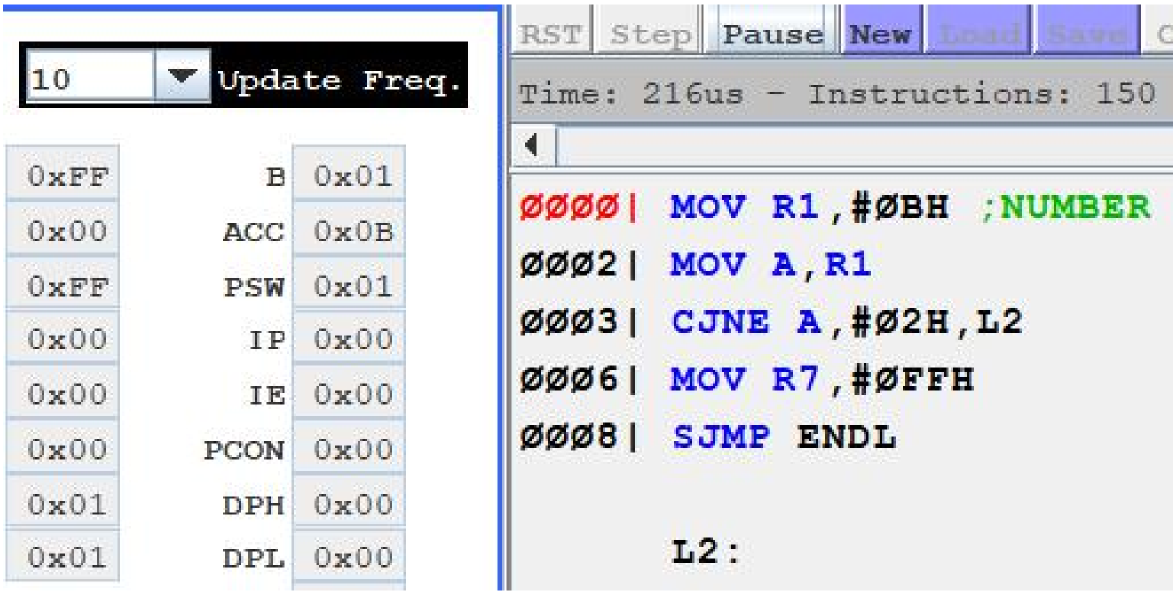
CODE:

```
MOV R1,#0BH ;NUMBER
MOV A,R1
CJNE A,#02H,L2
MOV R7,#0FFH
SJMP ENDL
L2:
DEC R1
CJNE R1,#01H,L3
MOV R7,#0FFH
SJMP ENDL
L3:
MOV B,R1
MOV R0,A
DIV AB
MOV A,R0
MOV R0,B
CJNE R0,#0H,L2
```

```
MOV R7,#0H

ENDL:
END
```

OUTPUT:

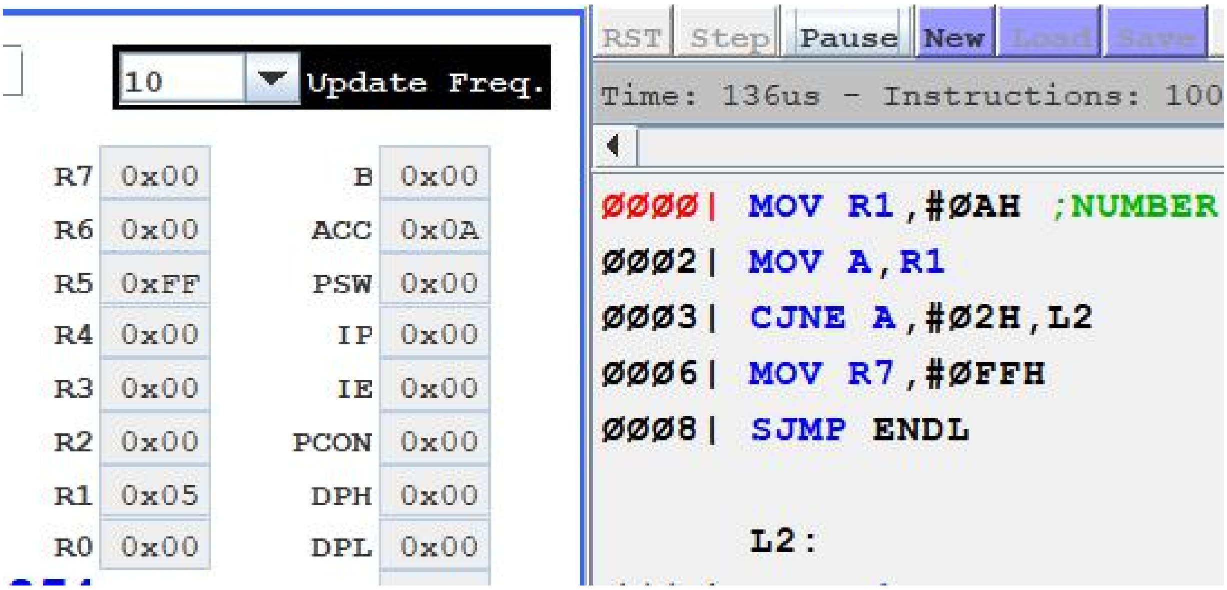
FOR PRIME NUMBER:



```
Time: 216us - Instructions: 150

0xFF        B    0x01
0x00        ACC  0x0B
0xFF        PSW  0x01
0x00        IP   0x00
0x00        IE   0x00
0x00        PCON 0x00
0x01        DPH  0x00
0x01        DPL  0x00
```

```
0000| MOV R1,#0BH  ;NUMBER
0002| MOV A,R1
0003| CJNE A,#02H,L2
0006| MOV R7,#0FFH
0008| SJMP ENDL

      L2:
```

FOR NON PRIME NUMBER:



```
Time: 136us - Instructions: 100

R7  0x00        B    0x00
R6  0x00        ACC  0x0A
R5  0xFF        PSW  0x00
R4  0x00        IP   0x00
R3  0x00        IE   0x00
R2  0x00        PCON 0x00
R1  0x05        DPH  0x00
R0  0x00        DPL  0x00
```

```
0000| MOV R1,#0AH  ;NUMBER
0002| MOV A,R1
0003| CJNE A,#02H,L2
0006| MOV R7,#0FFH
0008| SJMP ENDL

      L2:
```

5. Write an ALP to perform

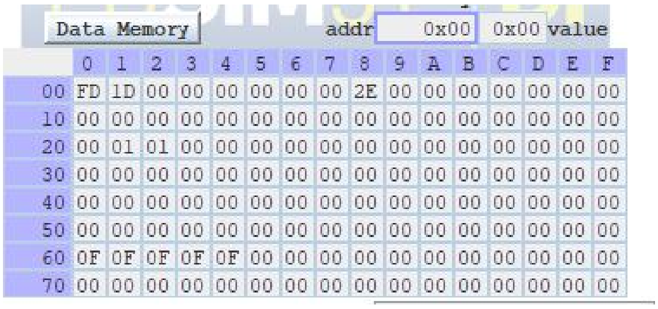
- **four different(any) conversion of number system**

1.DECIMAL TO BINARY:

```
MOV R7,#20
MOV R1,#23H
MOV R0,#4H
MAIN:
DEC R0
MOV A,R7
MOV B,#2
DIV AB
MOV R2,B
MOV B,#2
DIV AB
MOV R7,A
MOV A,B
MOV B,#10
MUL AB
ADD A,R2
DA A
MOV @R1,A
DJNZ R1,MAIN
END
```

OUTPUT:

8 bit representation for a given decimal number is store in address 20 t0 23.



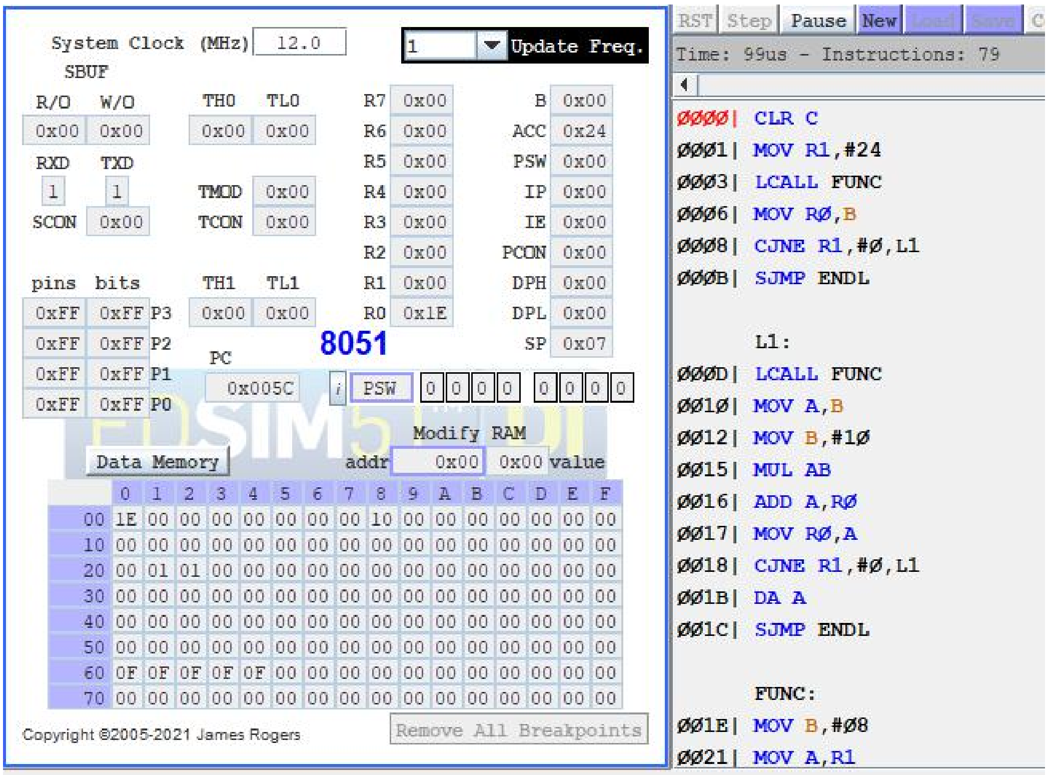
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	FD	1D	00	00	00	00	00	00	2E	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	01	01	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	0F	0F	0F	0F	0F	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

2. DECIMAL TO OCTAL:

```
CLR C
MOV R1,#24
LCALL FUNC
MOV R0,B
CJNE R1,#0,L1
SJMP ENDL
L1:
LCALL FUNC
MOV A,B
MOV B,#10
MUL AB
ADD A,R0
MOV R0,A
CJNE R1,#0,L1
DA A
SJMP ENDL
FUNC:
MOV B,#08
MOV A,R1
DIV AB
MOV R1,A
RET
ENDL:
END
```

OUTPUT:

(9)10 = (11)8



System Clock (MHz) 12.0 | 1 Update Freq.

SBUF R/O 0x00 W/O 0x00 | TH0 0x00 TL0 0x00
RXD 1 TXD 1 | TMOD 0x00 | TCON 0x00 | SCON 0x00

pins / bits: P3 0xFF 0xFF, P2 0xFF 0xFF, P1 0xFF 0xFF, P0 0xFF 0xFF | TH1 0x00 TL1 0x00 | PC 0x005C

R7 0x00, R6 0x00, R5 0x00, R4 0x00, R3 0x00, R2 0x00, R1 0x00, R0 0x1E

B 0x00, ACC 0x24, PSW 0x00, IP 0x00, IE 0x00, PCON 0x00, DPH 0x00, DPL 0x00, SP 0x07

8051 | PSW 0 0 0 0 0 0 0 0 | Modify RAM addr 0x00 0x00 value | Data Memory

```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 1E 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 00 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 0F 0F 0F 0F 0F 00 00 00 00 00 00 00 00 00 00 00
70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Copyright ©2005-2021 James Rogers | Remove All Breakpoints

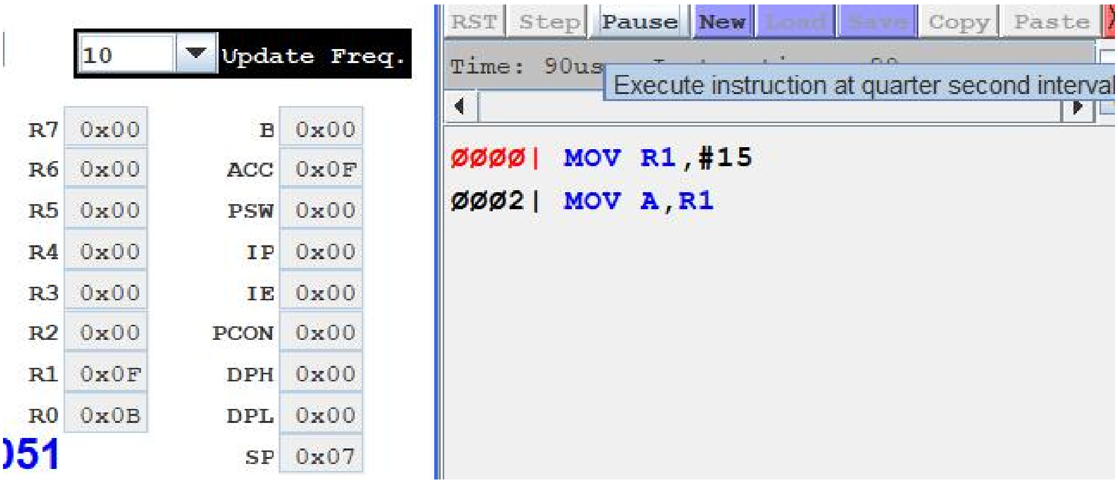
RST | Step | Pause | New | Load | Save

Time: 99us - Instructions: 79

```
ØØØØ| CLR C
ØØØ1| MOV R1,#24
ØØØ3| LCALL FUNC
ØØØ6| MOV RØ,B
ØØØ8| CJNE R1,#Ø,L1
ØØØB| SJMP ENDL

      L1:
ØØØD| LCALL FUNC
ØØ1Ø| MOV A,B
ØØ12| MOV B,#1Ø
ØØ15| MUL AB
ØØ16| ADD A,RØ
ØØ17| MOV RØ,A
ØØ18| CJNE R1,#Ø,L1
ØØ1B| DA A
ØØ1C| SJMP ENDL

      FUNC:
ØØ1E| MOV B,#Ø8
ØØ21| MOV A,R1
```

3. DECIMAL TO HEXADECIMAL

Output:



10 Update Freq.

R7 0x00, R6 0x00, R5 0x00, R4 0x00, R3 0x00, R2 0x00, R1 0x0F, R0 0x0B

B 0x00, ACC 0x0F, PSW 0x00, IP 0x00, IE 0x00, PCON 0x00, DPH 0x00, DPL 0x00, SP 0x07

)51

RST | Step | Pause | New | Load | Save | Copy | Paste

Time: 90us

Execute instruction at quarter second interval

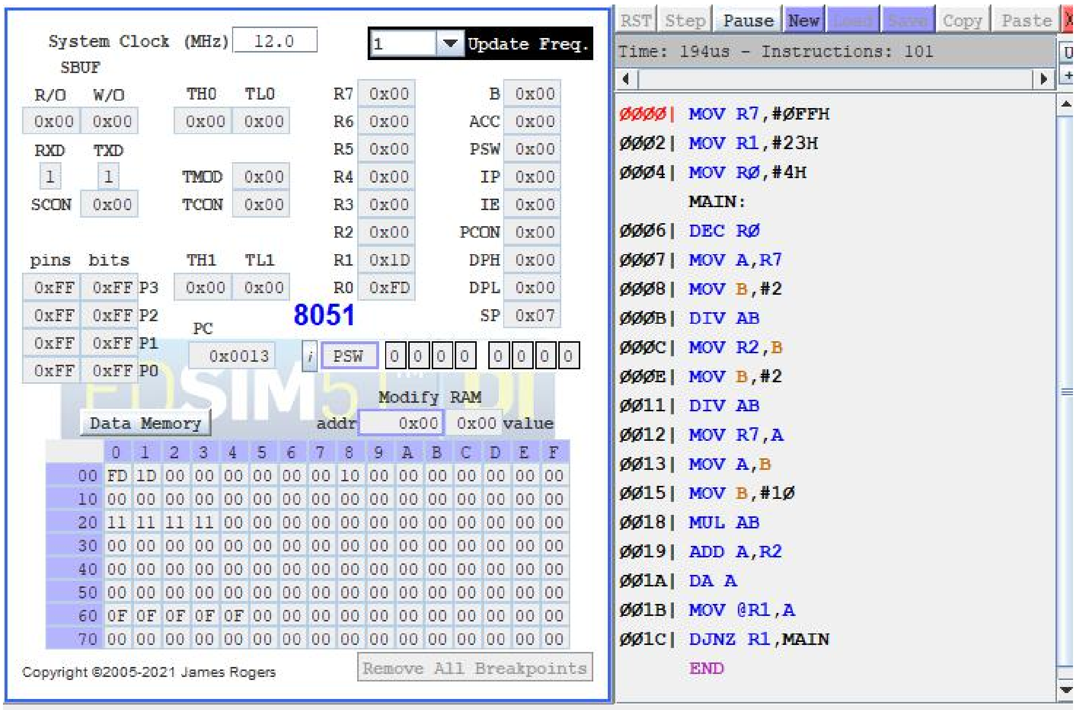
```
ØØØØ| MOV R1,#15
ØØØ2| MOV A,R1
```

4.HEXADECIMAL TO BINARY

```
MOV R7,#0FFH
MOV R1,#23H
MOV R0,#4H
MAIN:
DEC R0
MOV A,R7
MOV B,#2
DIV AB
MOV R2,B
MOV B,#2
DIV AB
MOV R7,A
MOV A,B
MOV B,#10
MUL AB
ADD A,R2
DA A
MOV @R1,A
DJNZ R1,MAIN
END
```

OUTPUT:

8 bit representation of given hexadecimal number



System Clock (MHz) 12.0

Update Freq. 1

```
SBUF
R/O   W/O      TH0   TL0      R7 0x00      B    0x00
0x00  0x00     0x00  0x00     R6 0x00      ACC  0x00
RXD   TXD                     R5 0x00      PSW  0x00
1     1        TMOD  0x00     R4 0x00      IP   0x00
SCON  0x00     TCON  0x00     R3 0x00      IE   0x00
                              R2 0x00      PCON 0x00
pins  bits     TH1   TL1      R1 0x1D      DPH  0x00
0xFF  0xFF P3  0x00  0x00     R0 0xFD      DPL  0x00
0xFF  0xFF P2                              SP   0x07
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x0013        PSW 0 0 0 0  0 0 0 0

Modify RAM
addr 0x00   0x00 value
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	FD	1D	00	00	00	00	00	00	10	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	11	11	11	11	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	0F	0F	0F	0F	0F	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2021 James Rogers

Time: 194us - Instructions: 101

```
ØØØØ| MOV R7,#ØFFH
ØØØ2| MOV R1,#23H
ØØØ4| MOV RØ,#4H
      MAIN:
ØØØ6| DEC RØ
ØØØ7| MOV A,R7
ØØØ8| MOV B,#2
ØØØB| DIV AB
ØØØC| MOV R2,B
ØØØE| MOV B,#2
ØØ11| DIV AB
ØØ12| MOV R7,A
ØØ13| MOV A,B
ØØ15| MOV B,#1Ø
ØØ18| MUL AB
ØØ19| ADD A,R2
ØØ1A| DA A
ØØ1B| MOV @R1,A
ØØ1C| DJNZ R1,MAIN
      END
```

\THANK YOU MAM !!\