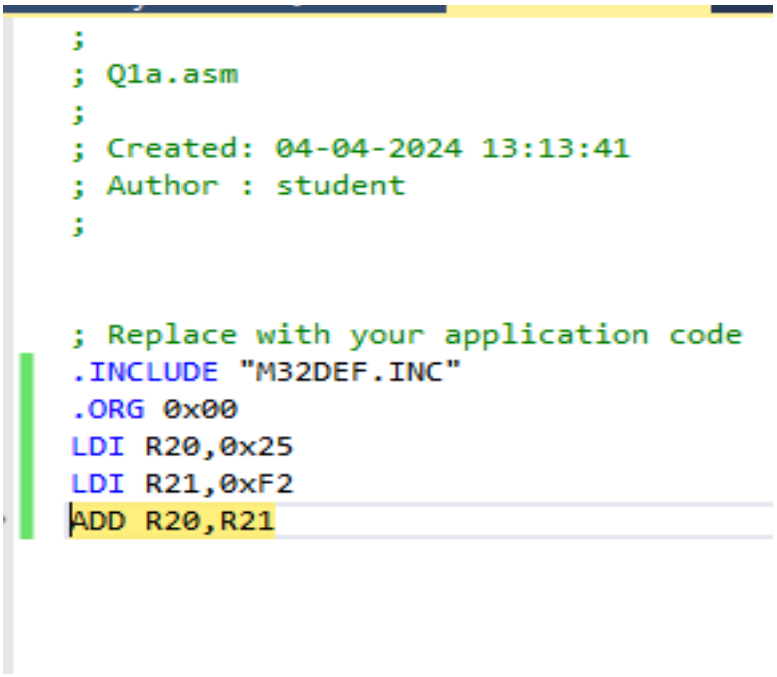
Q.1) Write assembly language program for AVR microcontroller to perform the following and analyse the contents of SREG register after each programming:

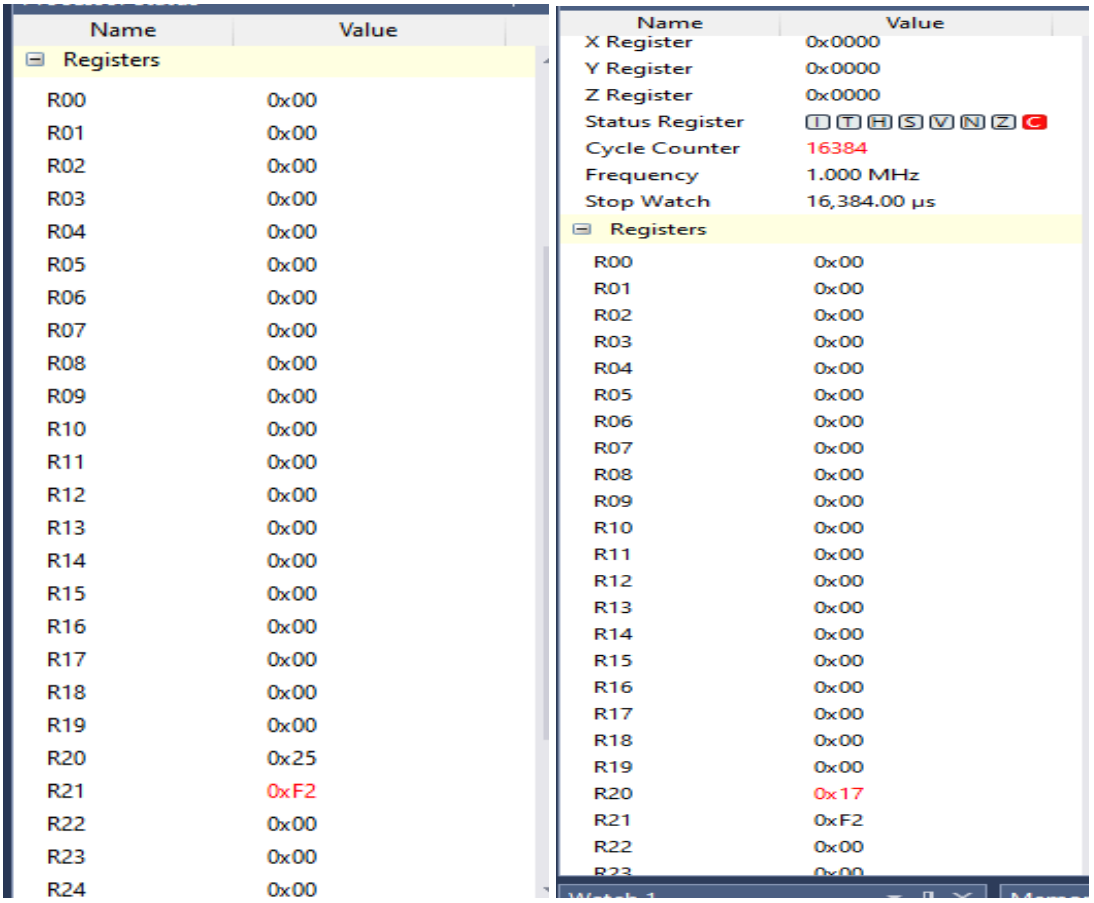
a) 25H + F2H

Code:

```
.INCLUDE "M32DEF.INC"
.ORG 0x00
LDI R20,0x25
LDI R21,0xF2
ADD R20,R21
```



OUTPUT



Name	Value
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x25
R21	0xF2
R22	0x00
R23	0x00
R24	0x00

Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	16384
Frequency	1.000 MHz
Stop Watch	16,384.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x17
R21	0xF2
R22	0x00
R23	0x00

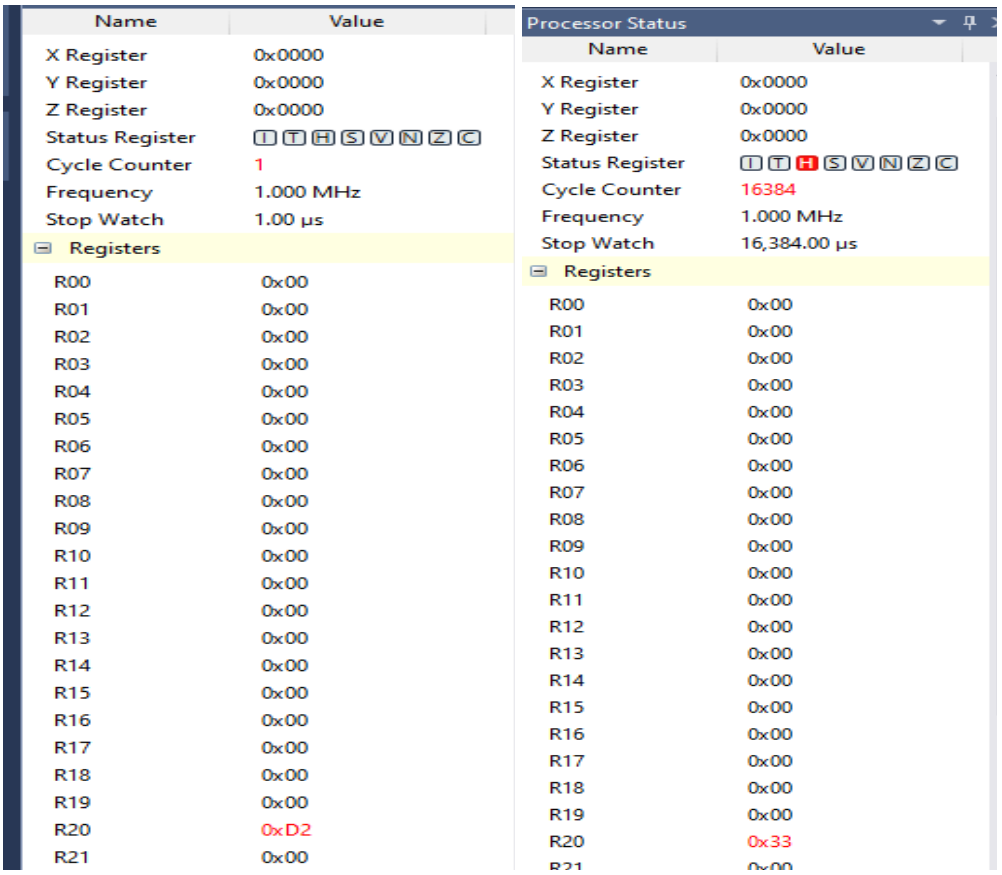
b) D2H-9FH

Code:

```
.INCLUDE "M32DEF.INC"
.ORG 0x00
LDI R20,0xD2
SUBI R20,0x9F
```

```
;
; Q1b.asm
;
; Created: 04-04-2024 13:21:21
; Author : student
;


; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0x00
LDI R20,0xD2
SUBI R20,0x9F
```

OUTPUT



Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	1
Frequency	1.000 MHz
Stop Watch	1.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0xD2
R21	0x00

Processor Status

Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	16384
Frequency	1.000 MHz
Stop Watch	16,384.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x33
R21	0x00

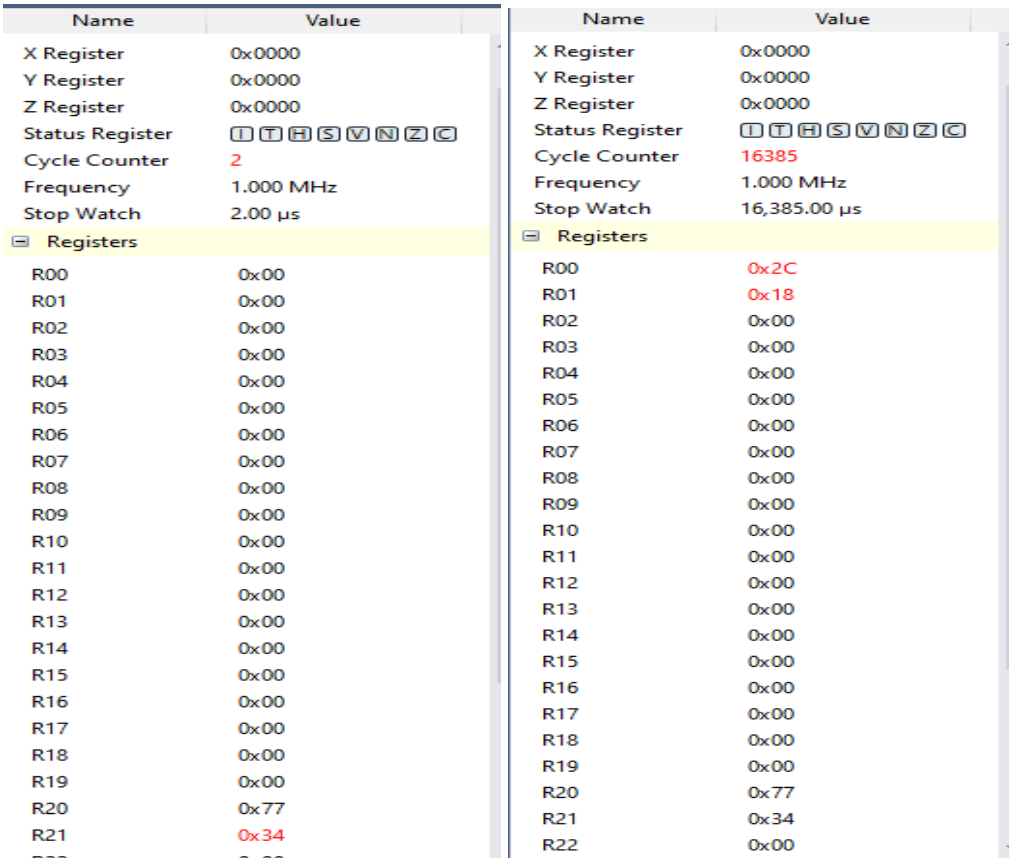
c) 0x77 x 0x34

Code:

```
.INCLUDE "M32DEF.INC"
.ORG 0x00
LDI R20,0x77
LDI R21,0x34
MUL R20,R21
```

```
;
; Q1b.asm
;
; Created: 04-04-2024 13:21:21
; Author : student
;


; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0x00
LDI R20,0x77
LDI R21,0x34
MUL R20,R21
```

OUTPUT



Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	2
Frequency	1.000 MHz
Stop Watch	2.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x77
R21	0x34

Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	16385
Frequency	1.000 MHz
Stop Watch	16,385.00 µs
Registers	
R00	0x2C
R01	0x18
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x77
R21	0x34
R22	0x00

d) 0x64/0x04

Code:

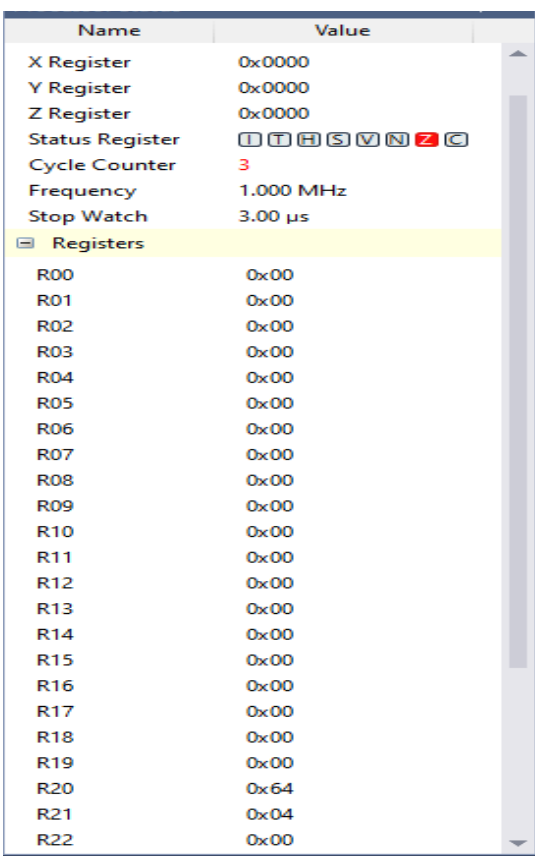
```
.INCLUDE "M32DEF.INC"
.ORG 0x0

.DEF NUM=R20
.DEF DENOMINATOR=R21
.DEF QUOTIENT=R22

LDI NUM,0X64
LDI DENOMINATOR,0X04
CLR QUOTIENT

L1: INC QUOTIENT
        SUB NUM,DENOMINATOR
        BRCC L1

        DEC QUOTIENT
```

ADD NUM,DENOMINATOR

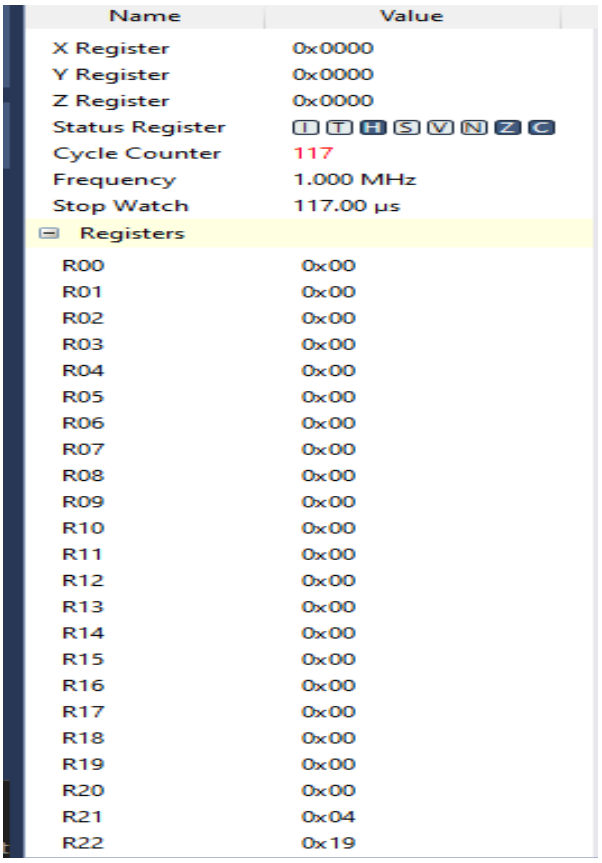
HERE: JMP HERE

```
; Created: 04-04-2024 13:21:21
; Author : student
;


; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0x0

.DEF NUM=R20
.DEF DENOMINATOR=R21
.DEF QUOTIENT=R22

LDI NUM,0X64
LDI DENOMINATOR,0X04
CLR QUOTIENT

L1: INC QUOTIENT
    SUB NUM,DENOMINATOR
    BRCC L1

    DEC QUOTIENT
    ADD NUM,DENOMINATOR

HERE: JMP HERE
```

OUTPUT



Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	3
Frequency	1.000 MHz
Stop Watch	3.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x64
R21	0x04
R22	0x00



Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	117
Frequency	1.000 MHz
Stop Watch	117.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x04
R22	0x19

e) AND 0x70, 0x6B

Code:

```
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R20,0X70
ANDI R20,0X6B
```

```
; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R20,0X70
ANDI R20,0X6B
```

OUTPUT

Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	1
Frequency	1.000 MHz
Stop Watch	1.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x70
R21	0x00
R22	0x00

Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	16384
Frequency	1.000 MHz
Stop Watch	16,384.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x60
R21	0x00
R22	0x00

f) XOR 0xAA, 0x95

Code:

```
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R20,0XAA
LDI R21,0X95
EOR R20,R21
```

```
; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R20,0XAA
LDI R21,0X95
EOR R20,R21
```

OUTPUT

Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	2
Frequency	1.000 MHz
Stop Watch	2.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0xAA
R21	0x95
R22	0x00

Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	16384
Frequency	1.000 MHz
Stop Watch	16,384.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x3F
R21	0x95
R22	0x00

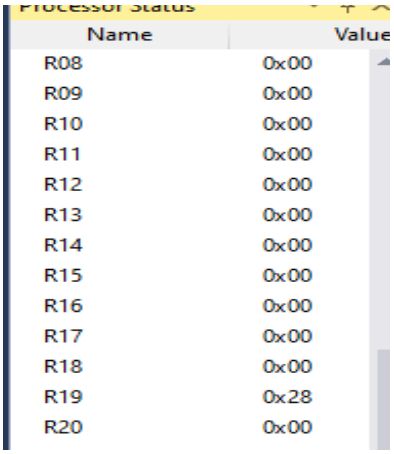
Q.2) Write assembly language program for AVR microcontroller to copy data from location $68 to Port D using R19.

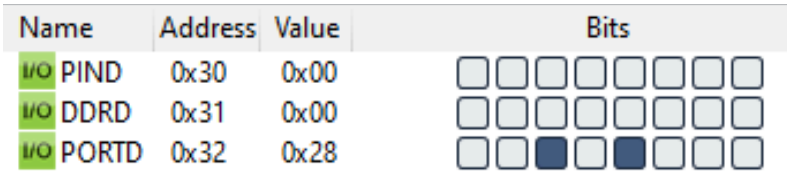
Code:

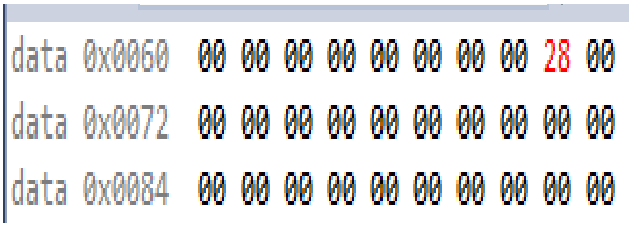
```
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R19, $28
STS $68, R19
LDS R19,$68
OUT PORTD, R19
```

```
; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R19, $28
STS $68, R19
LDS R19,$68
OUT PORTD, R19
```

OUTPUT







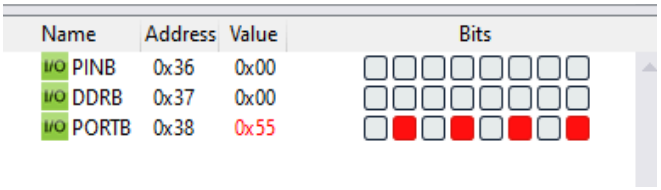
Q.3) Write assembly language program for AVR microcontroller to toggle all bits of PORTB by sending it to $55 and $AAH continuously. Put a time delay between each issuing of data to port B

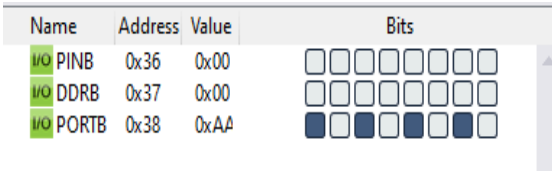
Code:

```
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16

BACK:
    LDI R16,0X55
    OUT PORTB,R16
    CALL DELAY
    LDI R16,0XAA
    OUT PORTB,R16
    CALL DELAY
    RJMP BACK

    .ORG 0X300
DELAY:
    LDI R20,0XFF
AGAIN:
    NOP
    NOP
    DEC R20
    BRNE AGAIN
    RET
```

```
; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16

BACK:
     LDI R16,0X55
     OUT PORTB,R16
     CALL DELAY
     LDI R16,0XAA
     OUT PORTB,R16
     CALL DELAY
     RJMP BACK

     .ORG 0X300
DELAY:
     LDI R20,0XFF
AGAIN:
     NOP
     NOP
     DEC R20
     BRNE AGAIN
     RET
```

OUTPUT





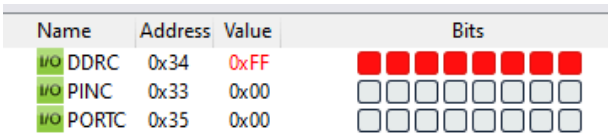
Q.4) An LED is connected to each pin of port C . Write an assembly program to turn on each pin from pin C0 to C7. Call a delay subroutine before turning on next LED.

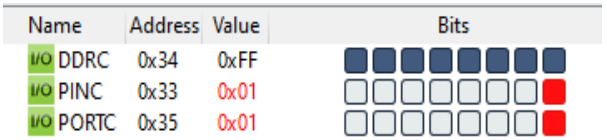
Code:

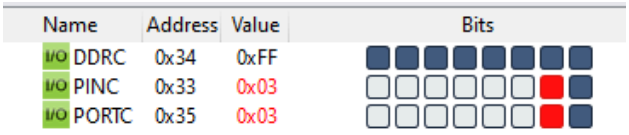
```
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16
LDI R16,0XFF
OUT DDRC,R16
SBI PORTC,0
CALL DELAY
SBI PORTC,1
CALL DELAY
SBI PORTC,2
CALL DELAY
SBI PORTC,3
CALL DELAY
SBI PORTC,4
CALL DELAY
SBI PORTC,5
CALL DELAY
SBI PORTC,6
CALL DELAY
SBI PORTC,7
CALL DELAY


.ORG 0X300
DELAY:
        LDI R20,0X05
AGAIN:
        NOP
        NOP
        DEC R20
        BRNE AGAIN
        RET
```

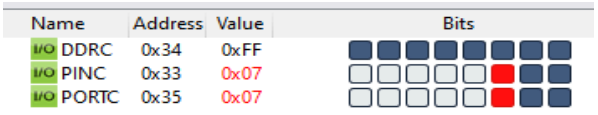
```
; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16
LDI R16,0XFF
OUT DDRC,R16
SBI PORTC,0
CALL DELAY
SBI PORTC,1
CALL DELAY
SBI PORTC,2
CALL DELAY
SBI PORTC,3
CALL DELAY
SBI PORTC,4
CALL DELAY
SBI PORTC,5
CALL DELAY
SBI PORTC,6
CALL DELAY
SBI PORTC,7
CALL DELAY
```

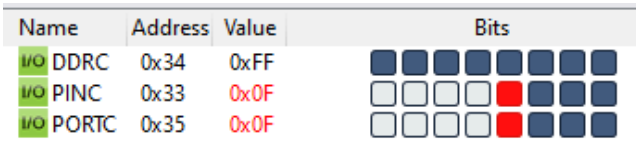
OUTPUT

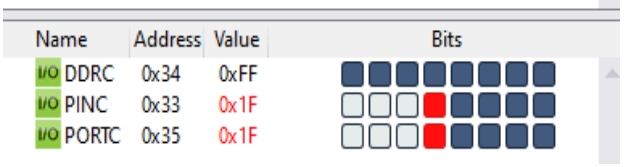


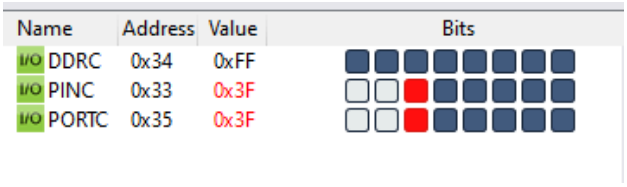


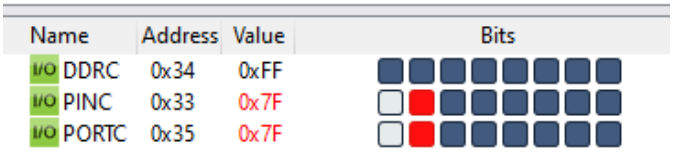


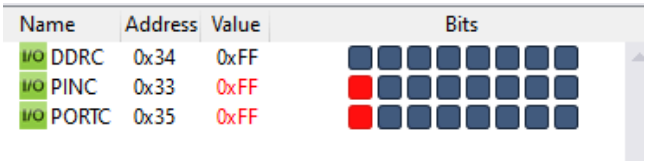












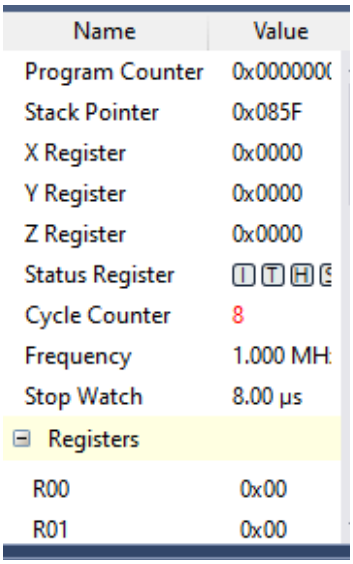
Q.5) Write an assembly program to create a square wave of 66% duty cycle on bit 2 of port C.

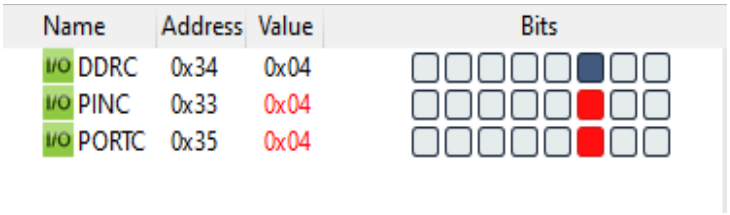
Code:

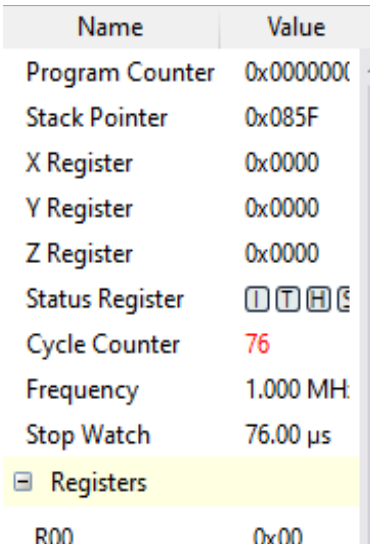
```
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16
SBI DDRC,2
HERE: SBI PORTC,2
CALL DELAY
CALL DELAY
CBI PORTC,2
CALL DELAY
RJMP HERE


.ORG 0X300
DELAY:
    LDI R20,0X05
AGAIN:
    NOP
    NOP
    DEC R20
    BRNE AGAIN
    RET
```

```
; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0x0

LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16
SBI DDRC,2
HERE: SBI PORTC,2
CALL DELAY
CALL DELAY
CBI PORTC,2
CALL DELAY
RJMP HERE


.ORG 0X300
DELAY:
    LDI R20,0X05
AGAIN:
    NOP
    NOP
    DEC R20
    BRNE AGAIN
    RET
```

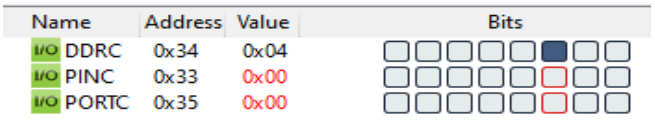
OUTPUT



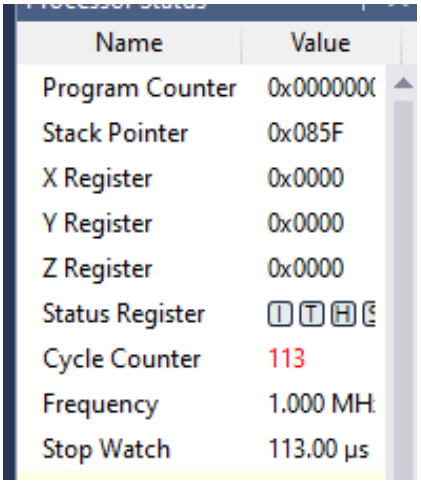




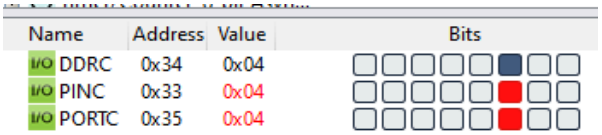
Name	Value
Program Counter	0x000000(
Stack Pointer	0x085F
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S
Cycle Counter	76
Frequency	1.000 MH
Stop Watch	76.00 µs
Registers	
R00	0x00



Name	Address	Value	Bits
DDRC	0x34	0x04	
PINC	0x33	0x00	
PORTC	0x35	0x00	



Name	Value
Program Counter	0x000000(
Stack Pointer	0x085F
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S
Cycle Counter	113
Frequency	1.000 MH
Stop Watch	113.00 µs



Name	Address	Value	Bits
DDRC	0x34	0x04	
PINC	0x33	0x04	
PORTC	0x35	0x04	

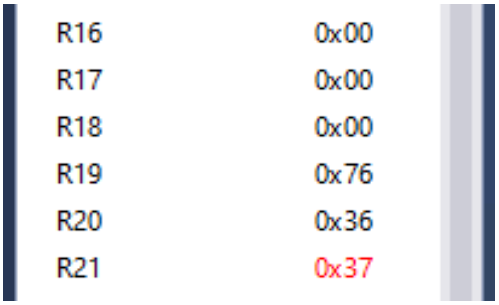
Q. 6) Write an assembly program to convert following packed BCD numbers to ASCII.
a) 0X76
b) 0X85 ,
Place the ASCII codes into R20 and R21.

a) Code:

```
LDI  R20,0x76  ;the packed BCD to be converted is 76
MOV  R19,R20  ;R20 = R19 = 76H
ANDI  R20,0x0F  ;mask the upper nibble (R20 = 06H)
ORI  R20,0x30  ;make it ASCII (R20 = 36H)
MOV  R21,R19  ;R21 = R19 = 76H
SWAP  R21  ;swap nibbles (R21 = 67H)
ANDI  R21,0x0F  ;mask the upper nibble (R21 = 07)
ORI  R21,0x30  ;make it ASCII (R21 = 37H)
```

```
;
; Q6.asm
;
; Created: 18-04-2024 13:15:00
; Author : student
;


; Replace with your application code
.INCLUDE "M32DEF.INC"
LDI   R19,0x76   ;the packed BCD to be converted is 76
MOV   R20,R19   ;R21 = R20 = 76H
ANDI  R20,0x0F   ;mask the upper nibble (R21 = 06H)
ORI   R20,0x30   ;make it ASCII (R21 = 36H)
MOV   R21,R19   ;R22 = R20 = 76H
SWAP  R21     ;swap nibbles (R22 = 67H)
ANDI  R21,0x0F   ;mask the upper nibble (R22 = 07)
ORI   R21,0x30   ;make it ASCII (R22 = 37H)
```

OUTPUT

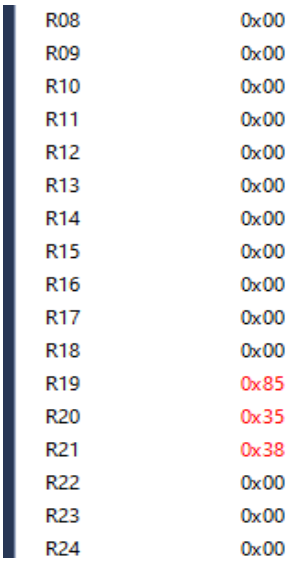


b) Code:

```
LDI  R20,0x85  ;the packed BCD to be converted is 85
MOV  R19,R20  ;R20 = R19 = 85H
ANDI  R20,0x0F  ;mask the upper nibble (R20 = 05H)
ORI  R20,0x30  ;make it ASCII (R20 = 35H)
MOV  R21,R19  ;R21 = R19 = 85H
SWAP  R21  ;swap nibbles (R21 = 58H)
ANDI  R21,0x0F  ;mask the upper nibble (R21 = 08)
ORI  R21,0x30  ;make it ASCII (R21 = 38H)
```

```
; Replace with your application code
LDI  R20,0x85  ;the packed BCD to be converted is 85
MOV  R19,R20  ;R20 = R19 = 85H
ANDI  R20,0x0F  ;mask the upper nibble (R20 = 05H)
ORI  R20,0x30  ;make it ASCII (R20 = 35H)
MOV  R21,R19  ;R21 = R19 = 85H
SWAP  R21  ;swap nibbles (R21 = 58H)
ANDI  R21,0x0F  ;mask the upper nibble (R21 = 08)
ORI  R21,0x30  ;make it ASCII (R21 = 38H)
```

OUTPUT



R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x85
R20	0x35
R21	0x38
R22	0x00
R23	0x00
R24	0x00

Q.7) Write an assembly program that finds number of zero's in a 8 bit data item.

Code:

```
.INCLUDE "M32DEF.INC"
.ORG 0x00
LDI R20,0b10111001
LDI R21, 8 ;set a counter to 8
LDI R22, 0 ;set number of zeros to 0
AGAIN:ROL R20 ;move MSB to Carry
BRCS NEXT ;branch if set carry
INC R22 ;increment number of zeros
NEXT: DEC R21 ;decrement the counter
BRNE AGAIN
```

```
; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0X00
LDI R20,0b10111001
LDI  R21, 8  ;set a counter to 8
LDI  R22, 0  ;set number of zeros to 0
AGAIN:ROL  R20    ;move MSB to Carry
BRCS  NEXT    ;branch if no carry
INC  R22    ;increment number of zeros
NEXT: DEC  R21    ;decrement the counter
BRNE AGAIN
```

OUTPUT

R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0xDC
R21	0x00
R22	0x03

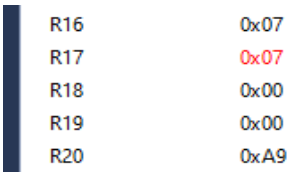
Q.8) Write an assembly program that finds position of first high in an 8 bit data item. The data is scanned from D7 to D0. Find the result for A9H.

Code:

```
.INCLUDE "M32DEF.INC"
.ORG 0x00
LDI R20, 0xA9
LDI R16,7
CLR R17

loop:
        SBRC R20,7
        RJMP found
        DEC R16
        ROL R20
        BRNE loop
        RJMP done

found:
        MOV R17,R16

done:
```

```
; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0x00
LDI R20, 0xA9   ; Load data into temporary register (A9H)
LDI R16, 7      ; Load 7 into register r16 (start from D7)
CLR R17         ; Clear r17 to store the position of the first high bit

loop:
    SBRC R20, 7     ; Check if the highest bit is set
    RJMP found       ; If set, jump to found
    DEC R16          ; Decrement the bit position
    ROL R20         ; Rotate data left through carry
    BRNE loop        ; If r16 != 0, continue looping
    RJMP done        ; If not found, jump to done

found:
    MOV R17, R16     ; Store the position of the first high bit in r17

done:
```

OUTPUT



R16	0x07
R17	0x07
R18	0x00
R19	0x00
R20	0xA9

Q.9) Assume that data memory location $240-$243 have following hex data. Write a program to add them together and place result in locations $220 and $221.
$240=$7D
$241=$EB
$242=$C5
$243=$5B

Code:

```
.INCLUDE "M32DEF.INC"
.ORG 0x00

LDI R19,0X7D
STS 0X240,R19
LDI R20,0XEB
STS 0X241,R20
LDI R21,0XC5
STS 0X242,R21
LDI R22,0X5B
STS 0X243,R22

.EQU L_BYTE = 0X220
.EQU H_BYTE = 0X221
LDI R16,4
LDI R20,0
LDI R21,0
LDI XL,0X40
LDI XH,0X02
L1:   LD R22,X+
      ADD R20,R22
      BRCC L2
      INC R21
      DEC R16
L2:   DEC R16
      BRNE L1
      STS L_BYTE,R20
      STS H_BYTE,R21
```

```
; Replace with your application code
.INCLUDE "M32DEF.INC"
.ORG 0x00

LDI R19,0X7D
STS 0X240,R19
LDI R20,0XEB
STS 0X241,R20
LDI R21,0XC5
STS 0X242,R21
LDI R22,0X5B
STS 0X243,R22

.EQU L_BYTE = 0X220
.EQU H_BYTE = 0X221
LDI R16,4
LDI R20,0
LDI R21,0
LDI XL,0X40
LDI XH,0X02
L1: LD R22,X+
    ADD R20,R22
    BRCC L2
    INC R21
    DEC R16
L2: DEC R16
    BRNE L1
    STS L_BYTE,R20
    STS H_BYTE,R21
```

OUTPUT

Registers

```
R00 = 0x00 R01 = 0x00
  R02 = 0x00 R03 = 0x00
  R04 = 0x00 R05 = 0x00
  R06 = 0x00 R07 = 0x00
  R08 = 0x00 R09 = 0x00
  R10 = 0x00 R11 = 0x00
  R12 = 0x00 R13 = 0x00
  R14 = 0x00 R15 = 0x00
R16 = 0x00 R17 = 0x00
  R18 = 0x00 R19 = 0x7D
  R20 = 0x88 R21 = 0x02
  R22 = 0x00 R23 = 0x00
  R24 = 0x00 R25 = 0x00
  R26 = 0x42 R27 = 0x03
  R28 = 0x00 R29 = 0x00
  R30 = 0x00 R31 = 0x00
```

```
data 0x01E6  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x020D  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 88 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0234  00 00 00 00 00 00 00 00 00 00 00 00 7d eb c5 5b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

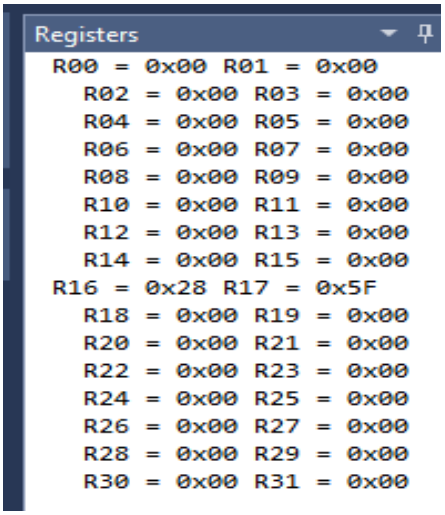
Q.10) Write an assembly program to read content of location 005F of EEPROM into portB.

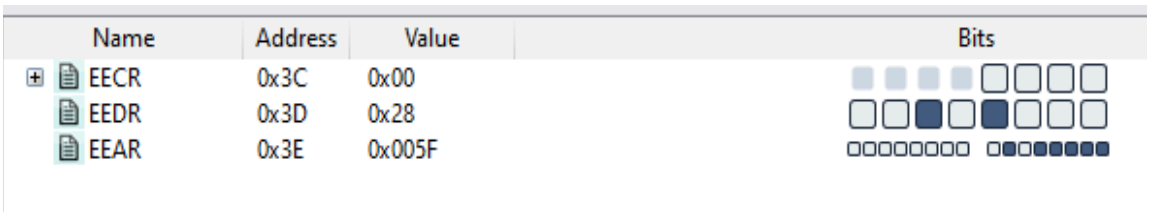
Code:

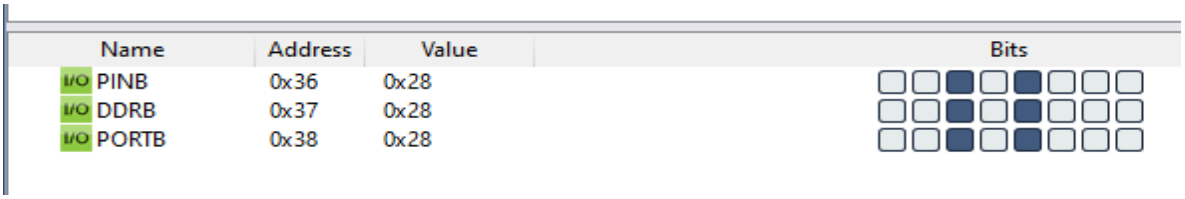
```
.INCLUDE "M32DEF.INC"

   .ORG 0x0000        ; Reset vector
   RJMP Main          ; Jump to the main program

Main:
   LDI R18, 0         ; Load 0 into R18
   LDI R17, 0x5F      ; Load the address 0x005F into R17

   OUT EEARH, R18     ; Set the EEPROM address high byte
   OUT EEARL, R17     ; Set the EEPROM address low byte

   LDI R16, 0x28      ; Load data (0xAA) into R16 to write to EEPROM
   OUT EEDR, R16      ; Write data to EEPROM data register

   SBI EECR, EEMWE    ; Enable EEPROM Master Write Enable
   SBI EECR, EEWE     ; Start EEPROM write by setting EEPROM Write Enable

WaitWrite:
   SBIC EECR, EEWE    ; Wait for write to complete
   RJMP WaitWrite     ; Loop back if write is still in progress

   SBI EECR, EERE     ; Initiate EEPROM read operation

   IN R16, EEDR       ; Read data from EEPROM into R16

   OUT DDRB, R16      ; Configure port B pins as outputs
   OUT PORTB, R16     ; Output the data to port B
```

```
; Replace with your application code
.INCLUDE "M32DEF.INC"

    .ORG 0x0000          ; Reset vector
    RJMP Main            ; Jump to the main program

Main:
    LDI R18, 0           ; Load 0 into R18
    LDI R17, 0x5F        ; Load the address 0x005F into R17

    OUT EEARH, R18       ; Set the EEPROM address high byte
    OUT EEARL, R17       ; Set the EEPROM address low byte

    LDI R16, 0x28        ; Load data (0xAA) into R16 to write to EEPROM
    OUT EEDR, R16        ; Write data to EEPROM data register

    SBI EECR, EEMWE      ; Enable EEPROM Master Write Enable
    SBI EECR, EEWE       ; Start EEPROM write by setting EEPROM Write Enable

WaitWrite:
    SBIC EECR, EEWE      ; Wait for write to complete
    RJMP WaitWrite       ; Loop back if write is still in progress

    SBI EECR, EERE       ; Initiate EEPROM read operation

    IN R16, EEDR         ; Read data from EEPROM into R16

    OUT DDRB, R16        ; Configure port B pins as outputs
    OUT PORTB, R16       ; Output the data to port B
```

OUTPUT







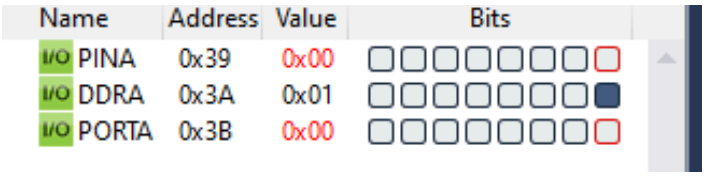
Q.11) Assuming XTAL=8MHz, write a C program to generate 1KHz frequency on PC4 using Timers.

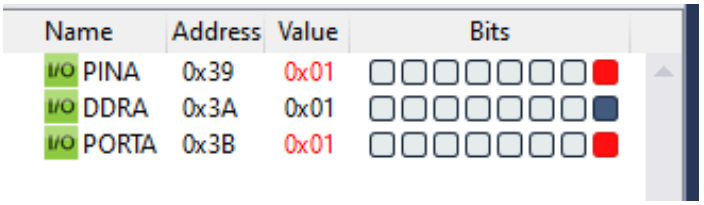
Code:

```
#include "avr/io.h"
void delay_500us ( );
int main ()
{
        DDRA = DDRA | 1;
        while(1)
        {
                PORTA = PORTA ^ (1 << PA0);
                delay_500us();
        }
        return 0;
}

void delay_500us ( )
{
        TCNT1H = (-500)>>8;
        TCNT1L = (-500)& 0xFF;
         TCCR1A = 0;
        TCCR1B = 0x2;
        TIFR = (1<<TOV1);
        while ((TIFR & (1<<TOV1)) == 0);
        TCCR1B = 0x00;
        TIFR = (1<<TOV1);
}
```

```c
#include "avr/io.h"
void delay_500us ( );
int main ()
{
    DDRA = DDRA | 1;
    while(1)
    {
        PORTA = PORTA ^ (1 << PA0);
        delay_500us();
    }
    return 0;
}


void delay_500us ( )
{
    TCNT1H = (-500)>>8;
    TCNT1L = (-500)& 0xFF;
    TCCR1A = 0;
    TCCR1B = 0x2;
    TIFR = (1<<TOV1);
    while ((TIFR & (1<<TOV1)) == 0);
    TCCR1B = 0x00;
    TIFR = (1<<TOV1);
}
```

OUTPUT

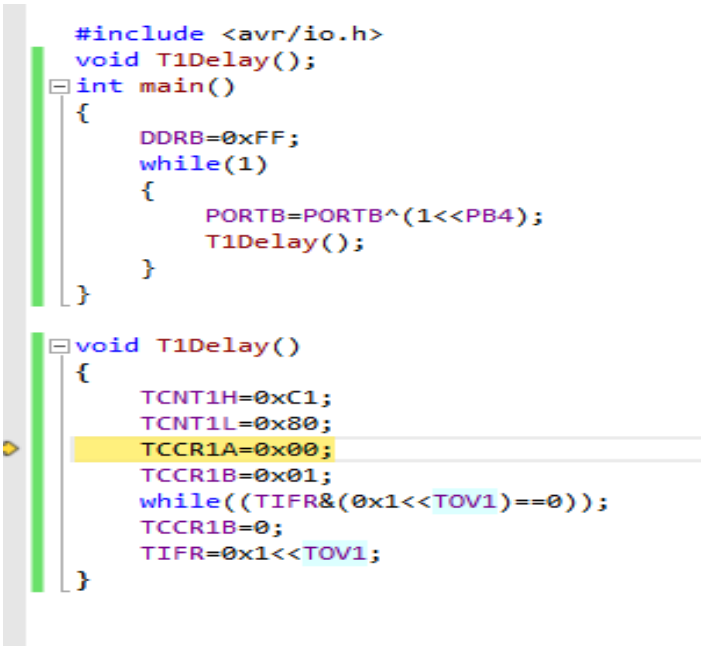




Q.12) Assuming XTAL=8MHz, write a C program to toggle portB.4 bit continuously every 2 ms. Use timer1, normal mode and no prescalar to create the delay.

Code:

```
#include "avr/io.h"
void T1Delay( );
int main ()
{
        DDRB = 0xFF;
        while(1)
        {
                PORTB = PORTB ^ (1 << PB4);
                T1Delay();
        }
}

void T1Delay( )
{
        TCNT1H = 0xC1;
        TCNT1L = 0x80;
         TCCR1A = 0x00;
        TCCR1B = 0x01;
        TIFR = (1<<TOV1);
        while ((TIFR & (1<<TOV1)) == 0);
        TCCR1B = 0x00;
        TIFR = 0x1<<TOV1;
}
```



OUTPUT

Name	Address	Value	Bits
I/O PINB	0x36	0x10	
I/O DDRB	0x37	0xFF	
I/O PORTB	0x38	0x10	

Name	Address	Value	Bits
ICR1	0x46	0x0000	
OCR1B	0x48	0x0000	
OCR1A	0x4A	0x0000	
TCNT1	0x4C	0xC182	
⊞ TCCR1B	0x4E	0x00	
⊞ TCCR1A	0x4F	0x00	
⊞ TIFR	0x58	0x00	
⊞ TIMSK	0x59	0x00	

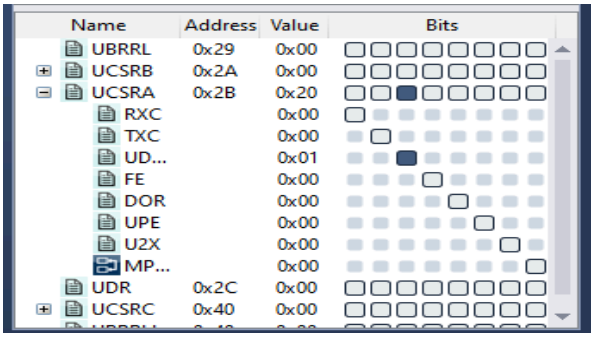
Name	Address	Value	Bits
I/O PINB	0x36	0x00	
I/O DDRB	0x37	0xFF	
I/O PORTB	0x38	0x00	

Q.13) Write a C program to send the message “IIOT is Industry 4.0” to the serial port continuously at 9600 baud rate. Use 8 bit data and 1 stop bit , assume XTAL=16MHz .

```
#include <avr/io.h>

void usart_init (void)
{
        UCSRB = (1<<TXEN);
        UCSRC = (1<<UCSZ1) | (1<<UCSZ0) | (1<<URSEL);
        UBRRL = 0x67;
}

void usart_send (unsigned char ch)
{
        while (! (UCSRA & (1<<UDRE)));
        UDR = ch;
}

int main (void)
{
        unsigned char str[20] = “IIOT is Industry 4.0”;
        unsigned char strLength = 20;
        unsigned char i = 0;
        usart_init();
        while(1)
        {
                usart_send(str[i++]);
                if (i >= strLength)
                        i=0;
        }
return 0;
}
```

```c
#include <avr/io.h>

void usart_init (void)
{
    UCSRB = (1<<TXEN);
    UCSRC = (1<<UCSZ1) | (1<<UCSZ0) | (1<<URSEL);
    UBRRL = 0x67;
}

void usart_send (unsigned char ch)
{
    while (! (UCSRA & (1<<UDRE)));
    UDR = ch;
}

int main (void)
{
    unsigned char str[20] = 'IIOT is Industry 4.0';
    unsigned char strLength = 20;
    unsigned char i = 0;
    usart_init();
    while(1)
    {
        usart_send(str[i++]);
        if (i >= strLength)
            i=0;
    }
return 0;
}
```

OUTPUT



Q.14) Using Timer0 and Timer1 interrupts generate a square wave on pins PB1 and PB7 respectively while transferring data from portC to portD assume XTAL=10MHz .

Code:

```
.INCLUDE "M32DEF.INC"
.ORG 0x0
        JMP MAIN
.ORG 0x12
        JMP T1_OV_ISR
.ORG 0x16
        JMP T0_OV_ISR

.ORG 0x100
MAIN:   LDI R20, HIGH(RAMEND)
          OUT SPH, R20
            LDI R20, LOW(RAMEND)
            OUT SPL, R20
            SBI DDRB, 1
            SBI DDRB, 7
            LDI R20, (1<<TOIE0) | (1<<TOIE1)
            OUT TIMSK, R20
            SEI
            LDI R20, 2   ; value for 20us
            OUT TCNT0, R20
            LDI R20, 0X01
            OUT TCCR0, R20
            LDI R20, HIGH(2)
            OUT TCNT1H, R20
            LDI R20, LOW(2)
            OUT TCNT1L, R20
            LDI R20, 0X00
            OUT DDRC, R20
            LDI R20, 0XFF
            OUT DDRD, R20

HERE:   IN R20. PINC
            OUT PORTD, R20
            JMP HERE

.ORG 0X200
T0_OC_ISR:
            LDI R16, 2
            OUT TCNT0, R16
```

```
        IN R16, PORTB
        LDI R17, 0X02
        EOR R16, R17
        OUT PORTB, R16
        RETI

.ORG 0X300
T1_OC_ISR:
        LDI R18, HIGH(2)
        OUT TCNT1H, R18
        LDI R18, LOW(2)
        OUT TCNT1L, R18
        IN R18, PORTB
        LDI R19, 0X80
        EOR R18, R19
        OUT PORTB, R18
        RETI
```

```
.INCLUDE "M32DEF.INC"
.ORG 0x0
    JMP MAIN
.ORG 0x12
    JMP T1_OV_ISR
.ORG 0x16
    JMP T0_OV_ISR

.ORG 0x100
MAIN:       LDI R20, HIGH(RAMEND)
             OUT SPH, R20
LDI R20, LOW(RAMEND)
OUT SPL, R20
SBI DDRB, 1
SBI DDRB, 7
LDI R20, (1<<TOIE0) | (1<<TOIE1)
OUT TIMSK, R20
SEI
LDI R20, 2   ; value for 20us
OUT TCNT0, R20
LDI R20, 0X01
OUT TCCR0, R20
LDI R20, HIGH(2)
OUT TCNT1H, R20
LDI R20, LOW(2)
OUT TCNT1L, R20
LDI R20, 0X00
OUT DDRC, R20
LDI R20, 0XFF
OUT DDRD, R20
```

```
HERE:    IN R20, PINC
         OUT PORTD, R20
         JMP HERE

.ORG 0X200
T0_OV_ISR:
         LDI R16, 2
         OUT TCNT0, R16
         IN R16, PORTB
         LDI R17, 0X02
         EOR R16, R17
         OUT PORTB, R16
         RETI

.ORG 0X300
T1_OV_ISR:
         LDI R18, HIGH(2)
         OUT TCNT1H, R18
LDI R18, LOW(2)
         OUT TCNT1L, R18
         IN R18, PORTB
         LDI R19, 0X80
         EOR R18, R19
         OUT PORTB, R18
         RETI
```

OUTPUT

Name	Address	Value	Bits
I/O PINB	0x36	0x00	☐☐☐☐☐☐☐☐
I/O DDRB	0x37	0x82	■☐☐☐☐☐■☐
I/O PORTB	0x38	0x00	☐☐☐☐☐☐☐☐

Q.14) Write a C program to interface LCD with AVR using 8 bit data.

Code:

```
#include <avr/io.h>
#include <util/delay.h>

#define LCD_DPRT PORTA
#define LCD_DDRA DDRA
#define LCD_DPIN PINA
#define LCD_CPRT PORTB
#define LCD_CDDR DDRB
#define LCD_CPIN PINB
#define LCD_RS 0
#define LCD_RW 1
#define LCD_EN 2

void delay_us(unsigned int d)
{
	delay_us(d);
}

void lcdCommand(unsigned char cmnd)
{
	LCD_DPRT = cmnd;
	LCD_CPRT &= ~(1<<LCD_RS);
	LCD_CPRT &= ~(1<<LCD_RW);
	LCD_CPRT |= (1<<LCD_EN);
	delay_us(1);
	LCD_CPRT &= ~(1<<LCD_EN);
	delay_us(100);
}

void lcdData(unsigned char data)
{
	LCD_DPRT = data;
	LCD_CPRT |= (1<<LCD_RS);
	LCD_CPRT &= ~(1<<LCD_RW);
	LCD_CPRT |= (1<<LCD_EN);
	delay_us(1);
	LCD_CPRT &= ~(1<<LCD_EN);
	delay_us(100);
}

void lcd_init()
{
```

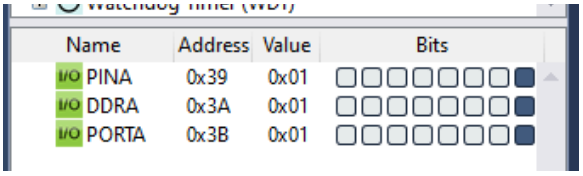
```
	LCD_DDDR = 0xFF;
	LCD_CDDR = 0xFF;

	LCD_CPRT &= ~(1<<LCD_EN);
	delay_us(2000);
	lcdCommand(0x38);
	lcdCommand(0x0E);
	lcdCommand(0x01);
	delay_us(2000);
	lcdCommand(0x06);
}

void lcd_gotoxy(unsigned char x, unsigned char y)
{
	unsigned char firstCharAdr[] = {0x80, 0xC0, 0x94, 0xD4};
	lcdCommand(firstCharAdr[y-1] + x -1);
	delay_us(100);
}

void_lcd_print(char * str)
{
	unsigned char i = 0;
	while(str[i] != 0)
	{
		lcdData(str[i]);
		i++;
	}
}

int main(void)
{
	lcd_init();
	lcd_gotoxy(1,1);
	lcd_print("The world is but");
	lcd_gotoxy(1,2);

	lcd_print("one country");

	while(1);
	return 0;
}
```

```
#include <avr/io.h>
#include <util/delay.h>

#define LCD_DPRT PORTA
#define LCD_DDDR DDRA
#define LCD_DPIN PINA
#define LCD_CPRT PORTB
#define LCD_CDDR DDRB
#define LCD_CPIN PINB
#define LCD_RS 0
#define LCD_RW 1
#define LCD_EN 2

void delay_us(unsigned int d)
{
    _delay_us(d);
}

void lcdCommand(unsigned char cmnd)
{
    LCD_DPRT = cmnd;
    LCD_CPRT &= ~(1<<LCD_RS);
    LCD_CPRT &= ~(1<<LCD_RW);
    LCD_CPRT |= (1<<LCD_EN);
_delay_us(1);
LCD_CPRT &= ~(1<<LCD_EN);
_delay_us(100);
}
```

```
void lcdData(unsigned char data)
{
    LCD_DPRT = data;
    LCD_CPRT |= (1<<LCD_RS);
    LCD_CPRT &= ~(1<<LCD_RW);
    LCD_CPRT |= (1<<LCD_EN);
_delay_us(1);
LCD_CPRT &= ~(1<<LCD_EN);
_delay_us(100);
}

void lcd_init()
{
    LCD_DDDR = 0xFF;
    LCD_CDDR = 0xFF;

    LCD_CPRT &= ~(1<<LCD_EN);
    _delay_us(2000);
    lcdCommand(0x38);
    lcdCommand(0x0E);
    lcdCommand(0x01);
    _delay_us(2000);
    lcdCommand(0x06);
}
```

```
void lcd_gotoxy(unsigned char x, unsigned char y)
{
    unsigned char firstCharAdr[] = {0x80, 0xC0, 0x94, 0xD4};
    lcdCommand(firstCharAdr[y-1] + x -1);
    _delay_us(100);
}

void lcd_print(char * str)
{
    unsigned char i = 0;
    while(str[i] != 0)
    {
        lcdData(str[i]);
        i++;
    }
}

int main(void)
{
    lcd_init();
    lcd_gotoxy(1,1);
    lcd_print('The world is but');
    lcd_gotoxy(1,2);

lcd_print('one country');

while(1);
return 0;
}
```

OUTPUT


Name
Address
Value
Bits
I/O PINA
0x39
0x01
I/O DDRA
0x3A
0x01
I/O PORTA
0x3B
0x01