**AVR Assembler programming Tutorial 5**

1. Write the AVR Assembly short code to configure **PORTB as output**?

LDI R16, 0xFF ; Set all PORTB pins to output

OUT DDRB, R16 ; Configure PORTB as output

2. Write the AVR Assembly short code **to turn on an LED connected to PORTB0**?

LDI R16, (1 << PB0) ; Load value to turn on PB0

OUT PORTB, R16 ; Set PORTB0 high

3. Write the AVR Assembly short code to turn off an LED connected to PORTB0?

LDI R16, ~(1 << PB0) ; Load value to turn off PB0

OUT PORTB, R16 ; Set PORTB0 low

4. Write the AVR Assembly short code to toggle an LED connected to PORTB0?

LDI R16, (1 << PB0) ; Load value for PB0

EOR PORTB, R16 ; Toggle PB0

5. Write the AVR Assembly short code to read the state of a switch connected to PORTA0?

IN R16, PINA ; Read PORTA

ANDI R16, (1 << PA0) ; Mask all but PA0

6. Write the AVR Assembly short code to perform addition of two numbers 3 and 5 stored in RAM locations 0x600 ?

LDI R16, 5 ; First number

LDI R17, 3 ; Second number

ADD R16, R17 ; R16 = R16 + R17

STS 0x600, R16

7. Write the AVR Assembly short code to subtract two numbers : 8 -3 stored in RAM locations 0x603 ?

LDI R19, 8 ; Minuend

LDI R20, 3 ; Subtrahend

SUB R19, R20 ; R19 = R19 – R20

STS 0x600, R19

8. Write the detail program AVR atmega32 assembly program to divide the two number 26/5 stored in R16 and R17 then store the result in memory locations 0x605 and 0x606?

LDI R16, 14 ; First number

LDI R17, 23 ; Second number

MUL R16, 17 ; R0:R1 = R16 \* R17 (result in R0:R1)

STS 0x608, R0

STS 0x609,R1

9. Write the AVR Assembly program in atmega32 microcontroller to divide the two number 26/5 stored in R16 and R17 then store the result in memory locations 0x605 and 0x606?

.include "m32def.inc" ; Include the definition file for ATmega32

.org 0x0000 ; Program start address

rjmp main ; Jump to main

main:

; Initialize values

LDI R16, 26 ; Dividend (26) - stored in R16

LDI R17, 5 ; Divisor (5) - stored in R17

LDI R18, 0 ; Initialize quotient (R18 = 0)

LDI R19, 0 ; Initialize remainder (R19 = 0)

divide\_loop:

; Check if the dividend (R16) is greater than or equal to the divisor (R17)

CP R16, R17 ; Compare R16 (dividend) with R17 (divisor)

BRLO store\_result ; If R16 < R17, go to store\_result

; Perform subtraction

SUB R16, R17 ; R16 = R16 - R17 (subtract divisor from dividend)

INC R18 ; Increment quotient (R18 = R18 + 1)

RJMP divide\_loop ; Repeat the loop

store\_result:

; Store the quotient and remainder

ST 0x605, R18 ; Store quotient at memory location 0x605

ST 0x606, R16 ; Store remainder (final value of R16) at 0x606

loop\_forever: ; End of program - loop forever

RJMP loop\_forever ; Infinite loop to end the program

10. Write the AVR Assembly code in ATMEGA32 to add 10 numbers stored in RAM locations 0x600 to 0x60A and output the result to PORTD?

LDI R19, 0xFF

OUT DDRD, R19 ; Config PORTD as 8 bit output

LDI R16, 0 ; Initialize sum to 0

LDI R18, 10 ; Number of elements

LDI R30, 0x00 ; LOW byte of Address of first number (ZL)

LDI R31, 0x06 ; HIGH byte Address of first number (ZH)

Loop:

LD R0, Z ; Load number from RAM

ADD R16, R0 ; Add to sum

INC Z ; Move to next number

DEC R18 ; Decrement counter

BRNE Loop ; Repeat until all numbers added

OUT PORTD, R16 ; output the Result R16 to PORTD

11. Write the AVR Assembly code in ATMEGA32 program that reads from PORTA input and performs an logic operation check PA0 pin, if PA0 pin is zero out 0x55 to PORTC otherwise out 0xAA to PORTC?

LDI R19, 0xFF

OUT DDRC, R19 ; Config PORTC as 8 bit output

IN R0, PINA ; Read input from PORTA

ANDI R0, 0x01 ; Mask to check first bit

BRNE go\_otherwise

LDI R17, 0x55

OUT PORTC, R17

go\_otherwise:

LDI R17, 0x55

OUT PORTC, R17

12. Write the AVR Assembly code in ATMEGA32 to read a byte from PORTB and store it in RAM location 0x603?

LDI R19, 0x00

OUT DDRB, R19 ; Config PORTB as 8 bit input

IN R0, PINB ; Read from PORTA

ST 0x603, R0 ; Store in RAM address 0x20

13. Write the AVR Assembly program that continuously reads a switch connected to PA0 and turn on a LED (connected to PB5) when switch state is 0, otherwise go to Loop back?

LDI R19, 0x00

OUT DDRB, R19 ; Config PORTA as 8 bit input

LDI R19, 0xFF

OUT DDRB, R19 ; Config PORTB as 8 bit output

Loop\_back:

IN R0, PINA ; Read switch state from PORTA

ANDI R0, (1 << PA0) ; Check if switch is pressed

BRNE Loop Loop\_back ; If not pressed, repeat

; Turn on LED

LDI R16, (1 << PB5) ; Load value to turn on PB0

OUT PORTB, R16 ; Set PORTB5 high