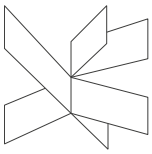
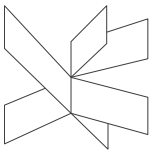


Mandatory Assignment

Table of content



1 Initialize port A as an output. Send 0xAA to port A

Solution

```
Ldi r16, 0xff
Out ddra, r16
Ldi r16 0xaa
Out porta, r16
```

2 Set bit 4 in ddra with out disturbing the remaining bits.

Solution

```
Sbi ddra, 4
```

3 Clear bit 3 in ddrb without disturbing the remaining bits.

Solution

```
Cbi ddrb, 3
```

4 Set bit 1,3,5 and 7 in DDRA without disturbing the remaining bits.

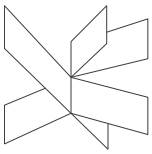
Solution

```
in r16, ddra
ori r16, 0b10101010
out ddra, r16
```

5 Clear bit 0, 1, 2 and 3 in DDRA without disturbing the remaining bits.

Solution

```
in r16, ddra
andi r16, 0b11110000
out ddra, r16
```



6 A switch is placed on PA3. If it is set then multiply r16 and r17 and send the result to portb (most significant byte) and portc (least significant byte). If PA3 is cleared add r16 and r17 and send the result to portb.

Solution

```
cbi ddra, 3 ; clear bit in ddra register to ensure that the switch is an input.
ldi r18, 0xff
out ddrb, r18 ; make this port an output.
out ddrc, r18 ; make this port an output.

sbic pina, 3 ; skip next instruction the bit is 0
rjmp bitcleared
add r16, r17
out portb, r16
rjmp forever

bitcleared:
mul r16, r17
out portb, r1
out portc, r0

forever:
rjmp forever ; stay here
```

7 Make a multiplier function using VIA calling convention

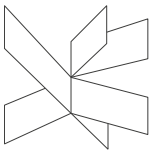
Solution

```
LDI R16, 0xFF; initializing the stack
OUT SPL, R16 ; initializing the stack
LDI R16, 0x21; initializing the stack
OUT SPH, R16 ; initializing the stack

ldi r26, 1 ; Original values of the working register (Should be the same after
the function has been executed)
ldi r27, 2
ldi r18, 3
ldi r19, 4

push r16 ; 1. r16 does not matter. This is for allocating place to the output
value
push r16 ; 1. r16 does not matter. This is for allocating place to the output
value

ldi r16, 10
```



```
ldi r17, 100
push r16 ; 1. Call setup
push r17 ; 1. CAll setup

call multiplierFunc ; 2. Call site
pop r17 ; 9. poping input values.
pop r16 ; 9.  poping input values.
pop r17 ; 9. Retrieving output value.
pop r16 ; 9. Retrieving output value.
stayhere:
rjmp stayhere

multiplierFunc:
push r0  ; 3. saving working registers
push r1  ; 3. saving working registers
push r26 ; 3. saving working registers
push r27 ; 3. saving working registers
push r18 ; 3. saving working registers
push r19 ; 3. saving working registers

in r26, SPL ; 4. (Retrienving input values). Setting op the X-register
in r27, SPH ;
adiw r26, 12 ; 6 pushes from working registers, 3 from return adress, 2 inputs,
and 1 ekstra.
LD R18, -X  ; 4. retrieving input value r16 = 10
LD R19, -X  ; 4. retrieving input value r17 = 100

mul r18, r19 ; 5. implementing the function body
adiw r26, 4  ; 5. updating the x-pointer to point at the right adress.
st -X,r0 ; 6. Saving output value 1
st -X,r1 ; 6. Saving output value 2

pop r19; 7. restoring working registers
pop r18; 7. restoring working registers
pop r27; 7. restoring working registers
pop r26; 7. restoring working registers
pop r1 ; 7. restoring working registers
pop r0 ; 7. restoring working registers

ret    ; 8. return from the function
```