USMAN BASHARAT
000874782

Logbook

Task 0

Examine the above program and complete the following table, showing the contents of each register after each line of the above program, is executed. Upload the grid and program flow chart to your logbook. This is predicted

	r16	r17	r18	r19
Line 1	04	-	-	-
Line 2	04	06	-	-
Line 3	04	06	01	-
Line 4	04	06	01	F8
Line 5	A	-	-	-
Line 6	9	-	-	-
Line 7	101	-	-	-

- FLOWCHART

```
    ldi  r16,$04   ;Line 1 - Put 04 HEX into register r16
    ldi  r17,$06   ;Line 2 - Put 06 HEX into register r17
    ldi  r18,$01   ;Line 3 - Put 01 HEX into register r18
    ldi  r19,$F8   ;Line 4 - Put F8 HEX into register r19
    add  r16,r17   ;Line 5 - Add contents of r16 to r17 and put the total in r16
    sub  r16,r18   ;Line 6 - Subtract the contents of r18 from the total in r16 and put the answer in
r16
    add  r16,r19   ;Line 7 - Add the contents of r16 to r19 and put the total in r16

end:  rjmp  end       ;loop forever
```

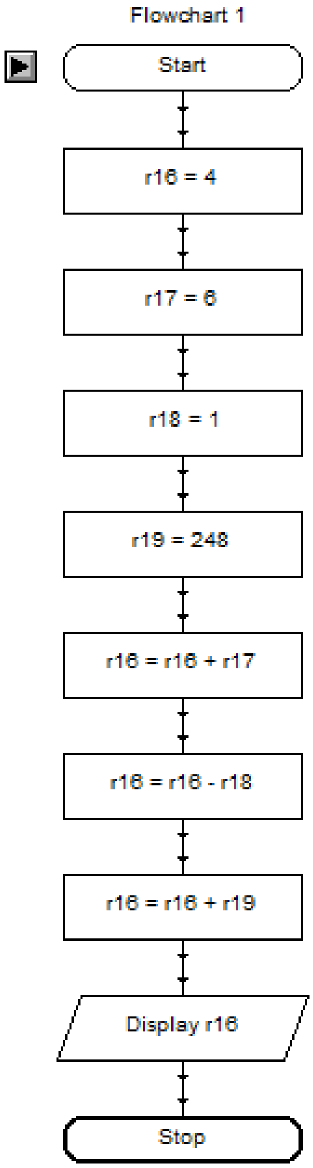
Task 1

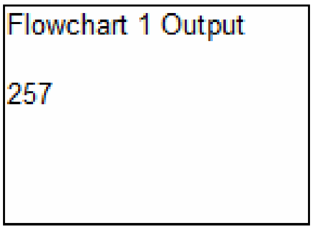
Single step (F10) through the program and complete the following table, showing the contents of each register after each line of the above program, is executed. Are the contents of the registers the same as you predicted in the table above? If not why not? Upload the grid and program flow chart to your logbook. This is when executed the program.

	r16	r17	r18	r19
Line 1	04	-	-	-
Line 2	04	06	-	-
Line 3	04	06	01	-
Line 4	04	06	01	F8
Line 5	A	06	01	F8
Line 6	09	06	01	F8
Line 7	01	06	01	F8

- FLOWCHART

USMAN BASHARAT
000874782





Flowchart 1 Output

```
257
```

Task 2

The following program should produce the answer to **3a + 2b - c** where **a=4**, **b=3**, **c=19**. Calculate what the answer should be. **Answer -1**

Task 3

Assemble the above program and correct the syntax errors. Explain the relationship of the answer produced by the simulator to the answer you calculated. Upload the corrected program and flow chart to your logbook.

Program to calculate 3a + 2b - c

```
.equ  a    =4

.equ  b    =3
```

USMAN BASHARAT
000874782

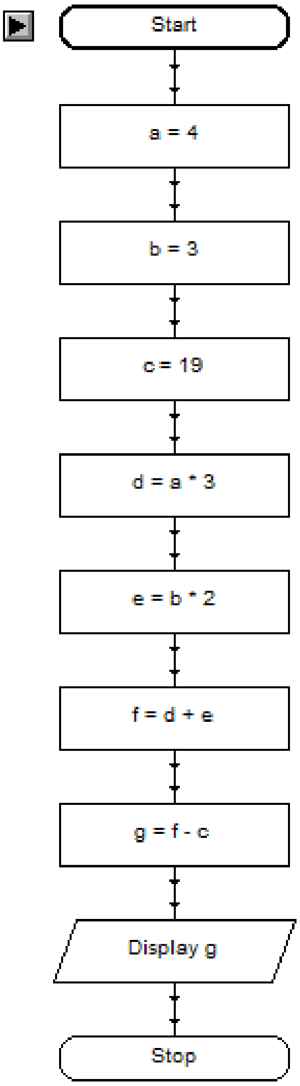
```
.equ   c     =19

    ldi   r16,a

    ldi   r17,b

    ldi   r18,c


;use register r20 to calculate 3a

    ldi   r20,$0

    add   r20,r16

    add   r20,r16

    add   r20,r16

;use register r21 to calculate 2b

    add   r21,r17

    add   r21,r17

;add 3a to 2b and put the result in r20

           add   r21,r20

           ldi   r20, $0

           mov   r20, r21

           ldi   r21, $0

;put c into r22 then take it from the total in r20

    ldi   r22,c

    sub  r20,r22


end:  rjmp  end     ;loop forever
```

ANSWER: -1   FF

- FLOWCHART

USMAN BASHARAT
000874782

Flowchart 1



Flowchart 1 Output

USMAN BASHARAT
000874782

Task 4

Assemble and single step the above program. Identify which flags are set in the status register and explain why the instruction / data caused the each of the flags to be set. Indicate which instructions do not effect the flags. Upload the completed table to your logbook. To understand the process that is occurring, you need to notate the values of r16 and r17 in binary.

Last two the same. 252 is FC in Hexadecimal and 249 is F9 in Hexadecimal. However, the flags, for the last code, has the same flags for all three lines.

;flag test program		r16	r17	I	T	H	S	V	N	Z	C	Explanation
ldi	r16,$80	10000000	-	-	-	-	-	-	-	-	-	Instruction does not effect flags
ldi	r17,$80	10000000	10000000	-	-	-	-	-	-	-	-	Instruction does not effect flags
add	r16,r17	00000000	10000000	0	0	0	1	1	0	1	1	Status: - Sign Flag, Twos Complement Overflow Flag, Zero Flag and Carry Flag. These flags are used to add those two numbers together.
Ldi	r16,$78	0111 1000 78	1000 0000 80	0	0	0	1	1	0	1	1	Instruction does not effect flags
Ldi	r17,$63	0111 1000 78	0110 0011 63	0	0	0	1	1	0	1	1	Instruction does not effect flags
Add	r16,r17	1101 1011 DB	0110 0011 63	0	0	0	0	1	1	0	0	Status: - Twos Complement Overflow Flag and Negative Flag
Ldi	r16,$FC	1111 1100 FC	0110 0011 63	0	0	0	0	1	1	0	0	Instruction does not effect flags. Status the same.
Ldi	r17,$F9	1111 1100 FC	1111 1001 F9	0	0	0	0	1	1	0	0	Instruction does not effect flags. Status the same.
Add	r16,r17	1111 0101 F5	1111 1001 F9	0	0	1	1	0	1	0	1	Status: - Half Carry Flag, Sign Flag, Negative Flag and Carry Flag.
Ldi	r16,252	1111 1100 FC	1111 1001 F9	0	0	1	1	0	1	0	1	Instruction does not effect flags. Status the same.
Ldi	r17,249	1111 1100 FC	1111 1001 F9	0	0	1	1	0	1	0	1	Instruction does not effect flags. Status the same.

Add	r16,r17	1111 0101 F5	1111 1001 F9	0	0	1	1	0	1	0	1	Status the same.

Task 5

Perform the following bitwise operations and upload the table to your logbook. To understand the process that is occurring, you need to notate the operand values in binary.

Operator	Operand 1	Operand 2	Answer
NOT	$A5	-	$90 (01011010)
AND	$A5 165	$0F	$5 (00000101)
OR	$A5 165	$0F	$175(1010 1111)
XOR	$A5 165	$0F	$170 (10101010)

Go over this again!

For an eight bit register

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Complete the following table, notate the operand values in binary and hexadecimal

	Operand 1	Operator	Operand 2	Answer
Flip bits 4 and 5	$A5 (1010 0101)	AND	1000 0101	$95 (1001 0101)
Extract bits 1 and 5	$A5 (1010 0101)	NOT	-	0101 1010
Set bits 4 and 6 to 1	$A5 (1010 0101)	XOR	65	228
	$A5 (1010 0101)	OR	$47 (0100 0111)	$E7(1110 0111)

USMAN BASHARAT
000874782

Do flowchart

Task 6

```
;bit shifting and rolling

.equ   a     =$FF


   ldi    r16,f

   ldi    r17,f

   ldi    r18,f

   ldi    r19,f


   lsl    r16

   lsl    r16

   lsl    r16

   lsl    r16

   com    r16

   andi   r16,$c0



   lsr    r17

   lsr    r17

   andi   r17,$30



   lsl    r18

   lsl    r18

   andi   r18,$c


   lsr    r19

   lsr    r19

   lsr    r19

   lsr    r19

   com    r19

   andi   r19,$3
   or    r16,r17

   or    r16,r18

   or    r16,r19

end:  rjmp   end
```

USMAN BASHARAT
000874782

Task 7

```
ldi r16,9

loop: dec r16

 dec r16

  brne loop

end: rjmp end
```

Execution Order	R16
1	09
2, 3, 4	08, 07, 06
5, 6, 7	05, 04, 03
8, 9, 10	02, 01, 00

```
        ldi     r16,0

loop:   inc     r16

        cpi     r16,4

        breq    loop

end:    rjmp    end
```

Execution Order	R16
1	01

Spins

```
        ldi     r16,1

loop:   inc     r16

        cpi     r16,5

        brne    loop
```

Execution Order	R16
1, 2	01, 02
3	03
4	04
5	05

```
        ldi     r16,3

loop:   inc     r16

        cpi     r16,6

        brne    loop
```

Execution Order	R16
1	03
2	04
3	05
4	06

This one only goes up to 6 and stops. Once it reaches 6, the loop stops. However, the difference between this one and the above is this one starts at 3 and goes up. However, the top one goes from one.

```
end:    rjmp    end
```

```
ldi     r16,1

loop:   inc     r16

        cpi     r16,3

        breq    next

next:   rjmp    loop
```

Execution Order	R16
1	01
2	02
3	03
4	04

This one goes in steps of adding each one again and again. This is the same as the first one instead it goes forward, not back. The first one and the last one does not stop. This is because it keeps adding one again and again to each other. It is like a forever loop.

```
end:    rjmp    end
```

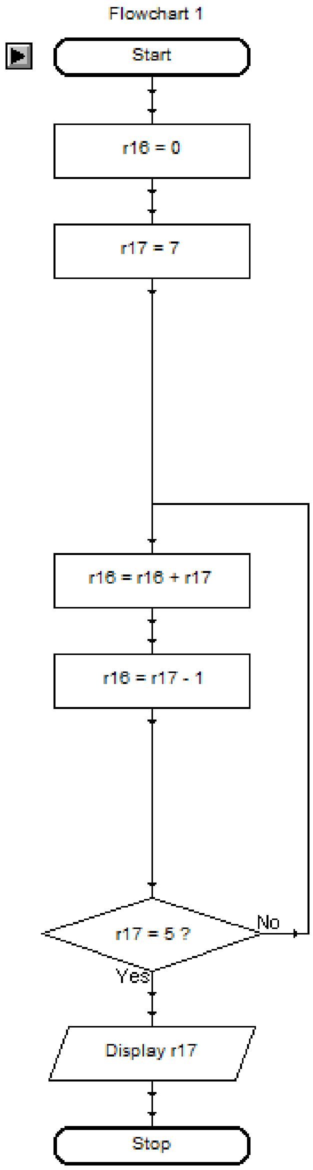
USMAN BASHARAT
000874782

Task 8

program to calculate 7+6+5+4+3+2+1 using a loop

```
              ldi   r16,0
               ldi   r17,7
loop: add   r16,r17
    dec  r17
    brne  loop


end:   rjmp    end       ;loop forever
```

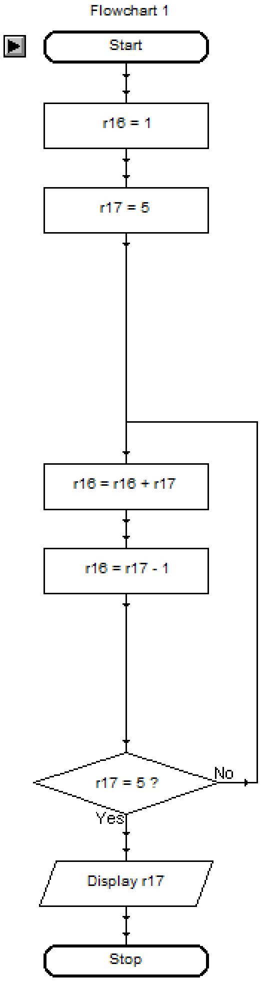
answer 1C



USMAN BASHARAT
000874782

```
;program to calculate 7+5+3+1 using a loop

    ldi   r16,1

    ldi   r17,5

loop: add   r16,r17

    dec   r17

    brne  loop


end:   rjmp    end       ;loop forever
```

answer 10 in hex



USMAN BASHARAT
000874782

Task 9

```
;Nested loop example

      ldi    r16,$28     ;Initialise counter

      ldi    r24,$5     ;Initialise 2nd loop counter

loop2: ldi    r25,$20    ;Initialise 1st loop counter

loop1: inc    r16         ;Increment counter

      dec    r25        ;Decrement the 1st loop counter

      brne   loop1       ;and continue to decrement until 1st loop counter = 0

      dec    r24        ;Decrement the 2nd loop counter

      brne   loop2       ;If the 2nd loop counter is not equal to zero repeat the 1st loop, else continue


end:   rjmp   end         ;loop forever
```

answer C8

Task 10 Reflection

It was all good. I felt that this was an exercise teaches how to use AVR programming language. I felt that Task 6 was the toughest out of all. I did the rest of them easily and understood this good. However, Task 6, I had to go over each of the notes to understand what is going on. After this, I understood that there was an not under bit 2, 3, 4 and 5. Once I did this, the tutor checked the code and got told that this was correct. I moved on to the other tasks and completed teach of them very quickly.