Лабораторна робота №1. Дослідження команд пересилання, арифметичних, логічних, роботи з окремими бітами та зсуву

Тема: команди пересилання даних, арифметичні, логічні, роботи з окремими бітами та зсуву.

Мета: користуючись налагоджувачем дослідити виконання команд пересилання даних, арифметичних, логічних, роботи з окремими бітами та зсуву у покроковому режимі.

Порядок виконання лабораторної роботи

1) Вивчити теоретичні відомості з теми «Команди пересилання даних, арифметичні, логічні, роботи з окремими бітами та зсуву». Дослідити формати (типи) команд, представлення операндів і роботу програми, що наведено у лабораторній роботі як приклад. Вміти коментувати команди, які наведено у таблиці 1.3.

2) Ввести програму з наведеними нижче командами пересилання даних у симулятор AVR-Studio 4.

3) Вивчити програмно-налагоджувальні засоби (ПНЗ) у AVR-Studio 4.

4) За допомогою ПНЗ проаналізувати виконання програми з наведеними нижче командами пересилання. Переконатися в правильному виконанні програми. При негативному результаті здійснити зміну програми та повторити перевірку.

5) Розробити алгоритм для виконання індивідуального завдання.

6) Розробити програму для виконання індивідуального завдання.

7) Ввести програму індивідуального завдання у симулятор AVR-Studio 4.

8) За допомогою ПНЗ проаналізувати виконання індивідуальної програми. Переконатися в правильному виконанні індивідуального завдання, при негативному результаті здійснити зміну алгоритму або програми, повторити перевірку.

9) Роздрукувати лістинг правильно працюючої програми.

10) Відповісти на контрольні питання викладача.

Команди пересилання даних

Стислі теоретичні відомості

Команди цієї групи призначено для пересилання вмісту комірок, що розташовані в просторі адрес статичної пам’яті даних (РЗП, РВВ та СОЗП). Поділ простору адрес на три частини (РЗП, РВВ, СОЗП) зумовлює різноманітність команд даної групи. Пересилання даних, яке виконується командами групи, може виконуватись в наступних напрямках:

– РЗП <=> РЗП;

– РЗП <=> РВВ;

– РЗП <=> статична пам’ять даних (РЗП, РВВ та СОЗП), 4 види адресації.

Також до даної групи можна віднести стекові команди PUSH і POP (відсутні в AT90S1200), що дозволяють зберігати в стеку і відновлювати зі стека вміст РЗП.

На виконання команд даної групи потрібно від одного до трьох машинних циклів в залежності від команди (таблиця 1.3).

Тип команди визначає як вона буде представлена у пам’яті програм процесора. Тип кожної команди зазначено в 6-му стовпці таблиці 1.3 і представлено на рисунку 1.1.

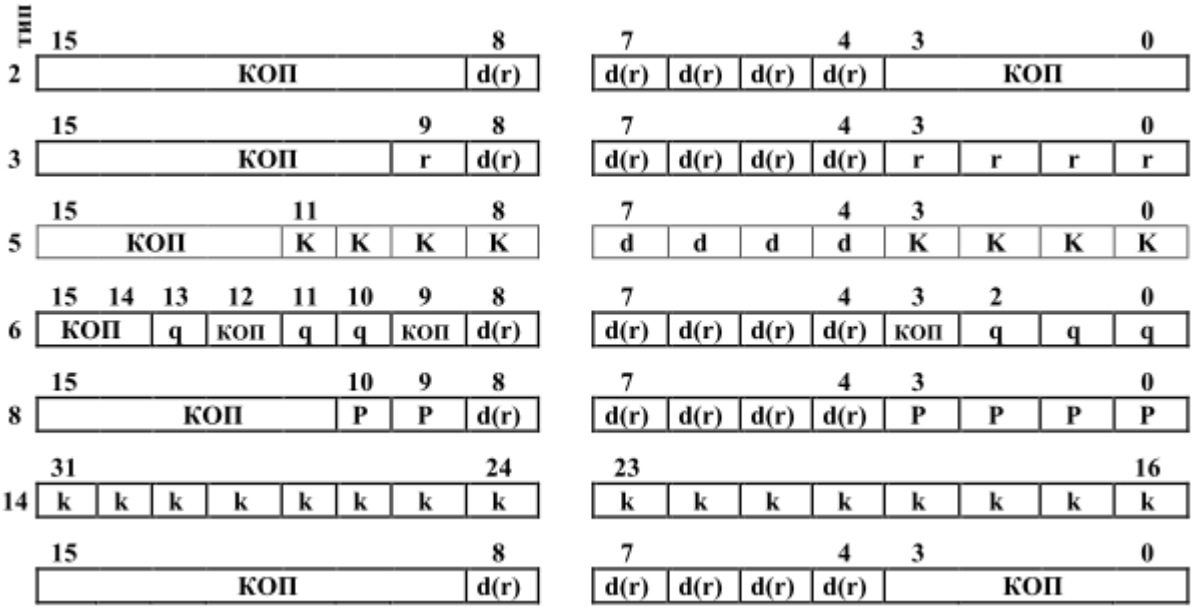

Рисунок 1.1 – Типи команд, що використовуються у командах пересилання даних

Нижче наведено приклад аналізу однієї з команд пересилання:

MOV Rd, Rr.

Опис: команда копіює вміст одного регістра в інший. Регістр джерело Rr залишається незмінним. Регістр приймач Rd завантажується копією регістра Rr. Номери регістрів Rr, Rd кодуються п’ятьма бітами і можуть приймати значення від 0 до 31 (2^5-1).

Приведена команда відповідає третьому типу (рис. 1.1), лише замість коду операції (КОП) компілятор підставляє конкретні цифри. КОП надає пристрою керування мікропроцесора інформацію які дії потрібно виконати при виконанні команди. $КОП_{команди}$ = 001011. Його можна визначити за допомогою симулятора AVR STUDIO4. Якщо створити проект, який складається лише з однієї команди MOV R1, R31, скомпілювати її, то область пам’яті програм матиме вигляд, який зображено на рис. 12.

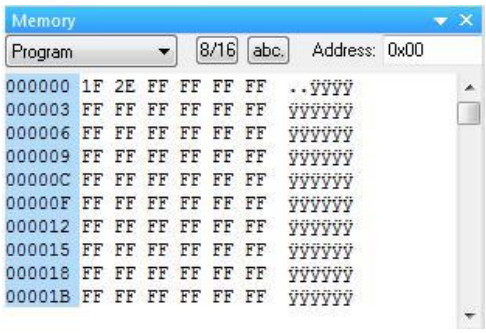

Рисунок 1.2 – Вигляд пам’яті програм після компіляції команди MOV R1, R31

Нижче наведено аналіз вмісту пам’яті програм.

Номера бітів	15							8	7							0
Машинний код команди MOV R1, R31	КОП						r	d	d	d	d	d	r	r	r	r
Код команди у двійковій формі	0	0	1	0	1	1	1	0	0	0	0	1	1	1	1	1
Код команди у шістнадцятковій формі	2			E				1				F				

Слід зазначити що в пам’яті програм спочатку розміщається другий байт команди, потім – перший: 2E 1F, у той час як в пам’яті програм бачимо запис: 1F 2E.

З отриманого машинного коду команди можемо визначити, що КОП = 001011, регістр-джерело r = $11111_2 = 31_{10}$, регістр-приймач d = $00001_2 = 1_{10}$.

Знайдені значення r та d дозволяють перевірити правильність розрахунків. Розглядалась команда MOV R1, R31. Розраховані значення r = 31, d = 1, робимо висновок, що розрахунки проведено вірно.

Довжина команди: 1 слово (2 байти).

Час виконання команди при тактовій частоті 12МГц дорівнює 1 такту або 1мкс.

Нижче наведено програму з командами пересилання даних.

Приклад програми з використанням команд пересилання даних

```
ldi R17, $32; Завантажити константу          R17 = $32
ldi R23, $45; Завантажити константу          R23 = $45
mov R17, R23 ; Копіювати регістр             R17 = R23
ldi R26, $60; Завантажити константу          R26 = $60
ldi R27, $00; Завантажити константу          R27 = $00
ld R3, X; Непряме завантаження               R3  = X
ld R4, X+; Непряме завантаження з постінкрементом R4    =X,X=X+1
ld R2, -X; Непряме завантаження з преддекрементом X=X-1,R2=X
ldi R28, $61; Завантажити константу          R28 = $61
ldi R29, $00; Завантажити константу          R29 = $00
ld R5, Y; Непряме завантаження               R5  = Y
ld R6, Y+; Непряме завантаження з постінкрементом R6    =Y,Y=Y+1
ld R7, -Y; Непряме завантаження з преддекрементом Y=Y-1,R7=Y
ldd R8, Y+2; Непряме завантаження зі зміщенням, в R8 завантажити
      ; вміст SRAM за адресою (Y + 2) = $63
ldi R30, $62; Завантажити константу          R30  = $62
ldi R31, $00; Завантажити константу          R31  = $00
ld R11, Z; Непряме завантаження              R11  = Z
ld R10, Z+; Непряме завантаження з постінкрементом      R10
      =Z,Z=Z+1
ld R12, -Z; Непряме завантаження з преддекрементом      Z=Z-1,R12=Z
ldd R18, Z+5; Непряме завантаження зі зміщенням R18     <=(Z+5)
lds R9, $001; Безпосереднє завантаження      R9 = $001
lds R10, $022; Безпосереднє завантаження     R10  = $022
lds R11, $080; Безпосереднє завантаження     R11  = $080
```

st X, R2; Непряме збереження X=R2
st X+, R10; Непряме збереження з постінкрементом X=R10,X=X+1
st -X, R11; Непряме збереження з преддекрементом X=X-1,X=R11
st Y, R7; Непряме збереження Y=R2
st Y+, R6; Непряме збереження з постінкрементом Y=R6,Y=Y+1
st -Y, R4; Непряме збереження з преддекрементом Y=Y-1,Y=R16
std Y+2,R16; Непряме збереження зі зміщенням (Y + 2) <= R16
st Z, R6; Непряме збереження Z=R6
st Z+, R7; Непряме збереження з постінкрементом Z=R7,Z=Z+1
st -Z, R17; Непряме збереження с преддекрементом Z=Z-1,Z=R17
std Z+4,R30; Непряме збереження зі зміщенням (Z + 4) <= R30
sts $82, R7; Безпосереднє збереження $82 <= R7
ldi R30, $01; Завантажити константу R30 = $01
lpm; Завантаження з пам’яті програм R0 = Z
in R2, UDR; Зчитування порта R2 <= UDR
out $16, R7; Запис у порт $16(PINB) <= R7
ldi R25, $DE; Завантажити константу R25 = $DE
out $3D, R25; Запис в порт $3D(SP) <= R25
push R4; Занесення регістра в стек STACK <= R4; SP = SP-1
pop R9; Витягнення регістра зі стека SP = SP+1, R9 <= STACK

Варіанти індивідуальних завдань

Таблиця 4.1 – Завдання до використання команд пересилання

№		Текст індивідуального завдання
1	a)	Вміст порту D (PIND) помістити в пам’ять даних за адресою 7F.
	b)	Використовуючи непряму адресацію, занести вміст R0, R2, R5 в пам’ять даних, починаючи з адреси 6E.
	c)	Вміст регістрів R0…R3 помістити в стек, попередньо встановивши вершину стека за адресою EF
2	a)	Вміст R5 записати в PORTD, використовуючи непряму адресацію та команду OUT.
	b)	У регістри R3, R4 завантажити два байти з пам’яті програм, починаючи з адреси 01H.
	c)	Зберегти вміст таймера/лічильника1 у пам’яті даних з адресою 6E, використовуючи непряму і безпосередню адресацію.
3	a)	Заповнити регістри R0…R7 вмістом комірок пам’яті даних, починаючи з 60H, використовуючи непряму адресацію.

	b)	Використовуючи команди роботи зі стеком, занести вміст пам’яті даних з адресою від 64H до 60H у довільно обрані регістри.
	c)	Записати число 23H в PORTD.
4	a)	Вміст молодшого байта таймера/лічільника1 занести в пам’ять даних за адресою80H.
	b)	Записати число 44H в PORTD використовуючи безпосередню адресацію, непряму адресацію і команди роботи зі стеком
	c)	Завантажити два байти в таймер/лічильник1 з пам’яті програм, починаючи з адреси06H.
5	a)	Вміст приймача/передавача (UDR) помістити в пам’ять даних.
	b)	Організувати під стек пам’ять даних з адреси 70H і помістити в стек вміст таймера/лічильника1 та порту D (PIND).
	c)	За адресою 60H в пам’яті даних знаходиться таблиця 3x5. Перші 3 елементи останнього рядка помістити в пам’ять даних, починаючи з адреси 80H.
6	a)	Обміняти вміст комірки пам’яті даних 74H та таймеру/лічильника 0.
	b)	Зберегти у стеку вміст приймача УАПП (UDR) та T/C1.
	c)	Діагональні елементи матриці 3x3, що розташована в пам’яті даних за адресою 80H, помістити в регістри R1…R5.
7	a)	Таймер/лічильник1 ініціалізувати значенням 4782D.
	b)	Записати вміст регістра R2 в регістр R3 не використовуючи команду MOV
	c)	Вміст регістрів R5…R7 помістити в стек, попередньо встановивши вершину стека за адресою AF.
8	a)	Вміст порту D (PIND) помістити в пам’ять даних за адресою 61H.
	b)	В регістри R5, R6 завантажити два байти з пам’яті програм, починаючи з адреси 05H.
	c)	Зберегти у стеку вміст порту D (PIND) та T/C1.
9	a)	Записати вміст регістра R5 в регістр R4 не використовуючи команду MOV.
	b)	Використовуючи команди роботи зі стеком, занести вміст пам’яті даних з адреси 64H до адреси 60H в довільно обрані регістри.
	c)	Записати число 1FH в T/C0 використовуючи бузпосередню адресацію, непряму адресацію і команди работи зі стеком.
10	a)	Організувати під стек пам’ять даних з адреси 80H і поместити в стек вміст таймера/лічильника1 та порту D (PIND).

	b)	Зберегти вміст таймера/ лічильника1 в пам’яті даних за адресою 6E, використовуючи непряму і безпосередню адресацію.
	c)	Записати число 3AH в PORTD.

Контрольні запитання

1. Які області пам’яті виділяють в програмістській моделі мікроконтролерів сімейства AVR?
2. Які операції пересилання даних дозволяє виконувати система команд мікроконтролерів AVR?
3. Який спосіб адресації використовується у цих командах?
4. Опишіть операнди, що входять до складу команд.
5. Звідки береться цифра 12 у формулі розрахунку часу виконання команди у секундах?

Команди арифметичні, логічні, зсуву та роботи з окремими бітами

Стислі теоретичні відомості

Команди даної групи наведено у таблиці 1.3. Формати команд наведено на рисунку 1.23, а їх опис – у 1.5.

Нижче наведено приклади арифметичних, логічних команд та команд зсуву.

Приклади арифметичних, логічних команд та команд зсуву

```
ldi R17, $32 ;Завантажити константу                  R17     = $32
ldi R23, $45 ;Завантажити константу                  R17     = $45
add R17,R23; Додавання без перенесення               R17     = R17 + R23
sec          ;Встановлення прапорця C                c = 1
adc R17,R23; Додавання с перенесенням                R17     =R17+R23+C
ldi R24, $15 ;Завантажити константу                  R24     = $15
ldi R25, $13 ;Завантажити константу                  R25     = $13
adiw R24, 62; Додавання до двох байт константи       ZH:ZL = ZH:ZL+62
sub R17, R23; Віднімання без перенесення             R17     = R17 - R23
subi R17,254; Віднімання константи                   R17         = R17 - 254
sbc R17, R23; Віднімання з перенесенням              R17         =R17-R23-C
sbci R17, 254; Віднімання константи з перенесенням          R12=R12-254-C
sbiw R24, 35; Віднімання з двох байт константи       ZH:ZL = ZH:ZL-35
```

and R17, R23; Логічне І	R17 = R17 * R23
andi R17, 230; Логічне І з константою	R17 = R17 * 230
or R23, R17; Логічне АБО	R23 = R23 v R17
ori R23, 45; Логічне АБО з константою	R23 = R23 v 45
eor R17, R23; Логічне виключне або	R17 = R17 ⊕ R23
com R17;Побітова інверсія	R17 = $FF - R17
neg R23;Зміна знака (додатковий код)	R23= $00 - R23
sbr R17, $55; Встановити парні біти в регістрі	R17 = R17 v $55
cbr R23, $AA; Скинути непарні біти в регістрі	R23 = R23 * ($FF - $AA)
inc R17; Інкремент значення регістра	R17 =R17+1
dec R23; Декремент значення регістра	R23 =R23-1
tst R17; Перевірка на нуль або від'ємність	R17 = R17 * R17
clr R23; Скинути регістр	R23 = $00
ser R17; Встановити регістр	R17 = $FF
swap R24; Обмін тетрадами регістра	R24
ldi R25, $05; Завантажити константу	R25 = $05
ldi R26, $C0; Завантажити константу	R26 = $C0
lsl R25; Логічний зсув ліворуч R25	
lsl R26; Арифметичний зсув вліво R26	
ldi R27, 12; Завантажити константу	R27 = 12
ldi R28, $F4 ; Завантажити константу	R28 = $F4
lsr R27; Логічний зсув праворуч R27	
asr R28; Арифметичний зсув праворуч R28	
ldi R29, $A5; Завантажити константу	R29 = $A5
rol R28; Циклічний зсув вліво R28	
ror R28; Циклічний зсув вправо R28	

Варіанти індивідуальних завдань до використання арифметичних, логічних команд та команд зсуву

Таблиця 4.2 – Завдання до використання арифметичних, логічних команд та команд зсуву

№		Текст індивідуального завдання
1	a)	Вміст регістра R3 і значення за адресою пам'яті даних 6Ch додати і помістити за адресою 7Ch
	b)	Додати двухбайтне число, що міститься в R3 і R4 і двухбайтне число, міститься в пам'яті даних 60h и 61h (попередньо помістивши туди значення).

	c)	Відняти від двухбайтного числа, що міститься в PORTC і PORTD, число 34 яке записано за адресою 0Dh і 1Dh попередньо помінявши в них тетради.
2	a)	Відняти 51 з регістра R3, результат помістити за адресою 8Ch.
	b)	Додати двобайтне число, що міститься за адресою 1Ch, 2Ch і двобайтне число F4.
	c)	Помножити двобайтне число на 4 і підрахувати кількість одиниці в отриманому результаті.
3	a)	Знайти суму 3-х членів ряда натуральних чисел, починаючи з числа #03d
	b)	Додати старшу і молодшу тетради регістра R26, результат записати в неупакованому форматі за адресою 70h.
	c)	Додати два числа розташованих за адресою 0Dh і 0Ch, результат помістити в стек, попередньо встановивши вершину стека за адресою EF
4	a)	Обчислити суму чисел -10 + 9 - …-2 +1.
	b)	Додати два однобайтних числа в прямом коді (7-й біт знаковий). Результат - в молодші тетради регістрів R26-R29
	c)	Дана матриця 1*3, розташована починаючи з адреси 60h. Знайти скільки елементів цієї матриці однакові з першим елементом. Результат записати в R29.
5	a)	Обчислити суму від’ємних чисел від -1 до -10.
	b)	Просумувати два числа, що розташовані за адресою 7Dh і 7Eh, результат помістити в регістр R29.
	c)	Знайти суму діагональних (головної і додаткової діагоналей) елементів матриці 2*2, що розташована починаючи з адреси 60h.
6	a)	Знайти суму цілих чисел, розташованих в діапазоні пам’яті даних, що обмежений вмістом PORTD і R0. (PORTD і R0 не мають нульових значень)
	b)	Додати два однобайтних числа в прямому коді(7-й біт знаковий). Результат помістити в пам’ять даних
	c)	Виконати циклічний зсув вліво двобайтного числа, записаного за адресою 7Dh і7Eh.
7	a)	Вміст PORTB і PORTC додати і помістити за адресою 70h в неупакованому форматі.
	b)	Додати двобайтне число, що розташовано в Т/Л1 і двобайтне число,

		розташоване в двох довільних регістрах
	c)	Відняти з двобайтного числа, розташованого в R23 і R24, число 34DEh, попередньо побітово проінвертувавши регістри.
8	a)	Додати дві двобайтні константи, використавши команду NEG.
	b)	Знайти суму 5623h і числа, розташованого в комірках 6Dh і 7Eh
	c)	Помістити масив 2x2, заповнений значеннями abs(i-j), де i – номер строки 0..1 і j – номер стовпця 0..1 починаючи з адреси 60h.
9	a)	Помножити двобайтне число, розташоване в R22, на 5.
	b)	В робочому регістрі R29 записано число в упакованому ДДК форматі. Додати тетради і результат помістить в пам’ять даних.
	c)	Додати два двійкові числа з цілими частинами в R12 і R13 і десятковою в R14 і R 15 відповідно.
10	a)	Обчислити різницю чисел 77h і EEh. Результат помножити на 2.
	b)	Обчислити кількість регістрів серед (R12-R16), в яких записано число 23h
	c)	Відняти від однобайтного числа двухбайтне.

Нижче наведено приклади команд роботи з бітами.

Приклади програми з використанням команд роботи з бітами

```
sbi $ 1C, 0; Встановити біт читання в EECR
cbi $ 12, 7; Скинути біт 7 у порту D

lsl R1; Логічний зсув вліво R1 (n +1) <= R1 (n), R1 (0) <= 0
lsr R1; Логічний зсув вправо  R1 (n) <= R1 (n +1), R1 (7) <= 0
rol R1; Циклічний зсув вліво через C R1 (0) <= C, R1 (n +1) <= R1 (n), C <= R1 (7)
ror R1; Циклічний зсув вправо через C R1 (7) <= C, R1 (n) <= R1 (n +1), C <= R1 (0)
asr R1; Арифметичний зсув вправо R1 (n) <= R1 (n +1), n = 0 .. +6

swap R2; Перестановка тетрад R1 (3 .. 0) <= R1 (7 .. 4), R1 (7 .. 4) <= R1 (3 .. 0)
bset 7     ; Встановлення прапорця SREG (s) <= 1 SREG (s)
bclr 7     ; Скидання прапорця SREG (s) <= 0 SREG (s)
bld R1, 7; Завантажити біт з T в регістр R1 (b) <= T
bst R1, 7 ; Зберегти біт з регістра в T, T <= R1 (b)
```

```
set     ; Встановити прапорець перенесення C <= 1
clc     ; Очистити прапорець перенесення C <= 0
sen     ; Встановити прапорець від'ємного числа N <= 1
cln     ; Скинути прапорець від'ємного числа N <= 0
sez     ; Встановити прапорець нуля Z <= 1
clz       ; Скинути прапорець нуля Z <= 0
sei       ; Встановити прапорець переривань I <= 1
cli       ; Скинути прапорець переривань I <= 0
ses       ; Встановити прапорець числа зі знаком S <= 1
cls       ; Скинути прапорець числа зі знаком S <= 0
sev       ; Встановити прапорець переповнення V <= 1
clv       ; Скинути прапорець переповнення V <= 0
set       ; Встановити прапорець T, T <= 1
clt       ; Скинути прапорець T, T <= 0
seh       ; Встановити прапорець допоміжного перенесення   H <= 1
clh       ; Скинути прапорець допоміжного перенесення H <= 0
nop       ; Немає операції
sleep     ; Спати (зменшити енергоспоживання)
wdr       ; Скидання вартового таймера
```

Варіанти індивідуальних завдань до використання команд роботи з бітами

Таблиця 4.3 – Завдання до використання команд роботи з бітами

№		Текст індивідуального завдання
1	a)	Скинути перший біт регістра керування таймера\лічильника 1 (TCCR1B).
	b)	Реалізувати комбінаційну двійкову функцію, задану аналітично: $bit5 = \overline{(bit1 * bit4)} + \overline{(bit1 * bit3 + bit3)}$
	c)	Реалізувати функцію задану у таблиці 4.4.
2	a)	Скинути перший біт регістра керування таймера\лічильника 0 (TCCR1B).
	b)	Реалізувати комбінаційну двійкову функцію, задану аналітично: $bit1 = \overline{R21(3) + R20(4)}$
	c)	Реалізувати функцію задану у таблиці 4.5
3	a)	Встановити третій біт у порту PORTB
	b)	Реалізувати комбінаційну двійкову функцію, задану аналітично: $bit1 = bit2 \vee \overline{(bit2 \wedge R23(2))} \wedge \overline{bit3}$

	c)	Реалізувати функцію задану у таблиці 4.6
4	a)	Скинути третій біт у порту PORTC.
	b)	: Знайти алгебраїчну суму бітів bit1 + bit2 + bit3 + bit4. Результат помістити у регістрі R7, у 3 молодших біта.
	c)	Реалізувати функцію задану у таблиці 4.7
5	a)	Переставити тетради регістра R3.
	b)	Реалізувати комбінаційну двійкову функцію, задану аналітично: $R11(3) = \overline{(bit1 * bit2 + bit2)} * bit3 + \overline{bit1}$
	c)	Реалізувати функцію задану у таблиці 4.8.
6	a)	Виконати циклічний зсув регістра R18 на 3 вправо.
	b)	Реалізувати комбінаційну двійкову функцію, задану аналітично: $bit1=bit2 \wedge \overline{(PORTD \wedge bit2)} \wedge bit3$
	c)	Реалізувати функцію задану у таблиці 4.9.
7	a)	Перші не зарезервовані 3 біта і останній біт з прямоадресуємої області помістити у старшу тетраду регістра R22.
	b)	Реалізувати логічну функцію $R13(0)=\overline{(bit1 * PORTC)} * \overline{R25(1)}$:
	c)	Реалізувати функцію задану у таблиці 4.5.
8	a)	Виконати логічний зсув регістра R17 на 5 вліво.
	b)	Реалізувати логічну функцію: $bit1=(\overline{bit1 \vee R22(2)} \vee bit2) \wedge \overline{bit1}$
	c)	Записати в область прямоадресуємих бітів перші 3 біта регістрів R22 и R23.
9	a)	Виконати арифметичний зсув регістра R17 на 5 вліво.
	b)	Реалізувати логічну функцію: $PORTB(2) = \overline{R12(7) + bit1}$
	c)	Просумувати по модулю 2 біти регістра R5. Результат помістити в прапорець додаткового перенесення.
10	a)	Проінвертувати перші 5 не зарезервованих біт з області прямоадресуємих біт і логічно підсумувати їх з 3-м бітом порту PORTC.
	b)	Реалізувати логічну функцію: $bit1=(bit1 \wedge bit2 \vee (bit1 \vee bit2)) \wedge bit1 \vee bit2 \vee bit3$
	c)	В R12 помістити перші вісім не зарезервованих прямоадресуємих біт.

Зауваження: bit, bit1, ... – будь-які біти з області прямоадресуємих біт.

Таблиця 4.4 – Булева функція 1

R17(0)	bit0	bit1	**F**
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Таблиця 4.5 – Булева функція 2

R17(0)	bit0	bit1	**F**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Таблиця 4.6 – Булева функція 1

R17(0)	bit0	bit1	**F**
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Таблиця 4.7 – Булева функція 2

R17(0)	bit0	bit1	**F**
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Таблиця 4.8 – Булева функція 1

R17(0)	bit0	bit1	**F**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Таблиця 4.9 – Булева функція 2

R17(0)	bit0	bit1	**F**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Контрольні запитання

1. Які арифметичні та логічні операції дозволяє виконувати система команд мікроконтролерів AVR?

2. Який спосіб адресації використовується у цих командах?
3. Як впливають арифметичні та логічні команди на прапорці?
4. Опишіть операнди, що входять до складу цих команд.
5. В якому поданні можуть бути використані числа при виконанні арифметичних та логічних операціях для мікроконтролерів AVR?
6. Які операції роботи з бітами дозволяє виконувати система команд мікроконтролерів AVR?
7. Які спосіби адресації використовуються у цих командах?
8. Опишіть операнди, що входять до складу цих команд.
9. Де знаходиться області прямоадресуємих біт?