Програмування в машинних кодах

Розглянемо різні способи завантаження даних до регістрів загального призначення:

- пряме завантаження

  LDS R2, $DF; запис до R2 вмісту комірки $00DF;

- завантаження константи

  LDI R16, $23; запис до R6 числа $23;

- завантаження з програмної пам'яті

```
CLR R31
LDI R30, $F0    ;записати в регістрову пару Z адресу $00F0
LPM             ;записати вміст комірки з адресою $F0 в R0;
```

- непряме завантаження, непряме завантаження з постінкрементом, з предекрементом

```
CLR R27
LDI R26, $20    ;записати в регістрову пару X адресу $0020
LD R0, X+       ;завантажити в R0 вміст SRAM за адресою $20 (X+1
                постінкремент)
LD R1, X        ;завантажити в R1 вміст за адресою $21
LDI R26, $23    ;записати в регістрову пару X адресу $0023
LD R3, -X       ;завантажити в R3 вміст SRAM за адресою $22 (X-1
                ;предекремент)
```

Розглянемо особливості програмування на мові Асемблер на прикладах.

Приклад 1

Заповнити область пам'яті SRAM з $DF по $BF кодом $03

```
    CLR R31         ;очистити регістр R31
    LDI R30, $DF    ;записати в регістрову пару Z адресу $DF
    LDI R20, $03    ;завантажити в R20 константу $03
L1: ST Z+, R20      ;зберегти число з R2 по адресі в регістровій парі Z з
                    ;послідуючим збільшенням вмісту регістрової пари
    CPI R30, $C0    ;порівняти вміст R30 з константою $C0
    BRNE L1:        ;перейти до мітки L1: якщо не рівні значення
```

Приклад 2

Заповнити область SRAM $60-$DF числами від 128 до 0.

```
    CLR R31         ;настройка регістрової пари Z, ZH=0
    LDI R30, $60    ;настройка регістрової пари Z, ZL=$60
    LDI R16, 128    ;в регістр R16 записуємо число 128
L1: ST z+, R16      ;число з R16 записується до комірки SRAM, адреса якої
                    ;знаходиться у регістровій парі z і збільшується  зміст z на 1
```

```
        DEC R16         ;відняти 1 з R16
        BRNE L1         ;перевірити чи  не записали всі числа (R16=0), якщо ні то
                        ;повторяємо процедуру запису числа
```

Приклад 3

Заповнити область пам'яті SRAM з $60…$A0 числом $A0, а з $A1 по $DF числом з $55

```
        CLR R31         ;очистити регістр R31
        LDI R30, $60    ;записати в регістрову пару Z адресу $60
        LDI R20, $0A    ;завантажити в R2 константу $0A
L1:     ST X+, R20      ;зберегти число з R20 по адресі в регістровій парі X з
                        ;послідуючим збільшенням вмісту регістрової пари
        CPI R30, $A1    ;порівняти вміст R30 з константою $A1
        BRNE L1:        ;перейти до мітки L1: якщо не рівні значення
        LDI R20, $55    ;завантажити в R20 константу $55
L2:     ST X+, R20      ;зберегти число з R20 по адресі в регістровій парі X з
                        ;послідуючим збільшенням вмісту регістрової пари
        CPI R30, $DF    ;порівняти вміст R30 з константою $DF
        BRNE L2:        ;перейти до мітки L2: якщо не рівні значення
```

Приклад 4

В області SRAM $60-$DF перевірити чи є запис числа $0C, якщо є замінити їх на $AF, а в регістр R10 записати кількість замінених байтів.

```
        CLR R31         ; запис до регістрової пари Z адреси першого числа
        LDI R30, $60
        CLR R10         ; очищення лічильника замінених чисел
        LDI R16, $AF    ; запис числа заміни
L2:     LD R17, Z       ; зчитати перше число
        CPI R17, $0C    ; порівняти його з $0C
        BRNE L1         ; якщо не дорівняє перейти по L1, інакше
        ST Z, R16       ; замість $0C записати $AF
        INC R10         ; збільшуємо лічильник кількості замін
L1:     INC R30         ; перехід до нової комірки пам'яті
        CPI R30 , $E0   ; порівнюємо, чи не досягли кінця масиву
        BRNE L2         ; якщо не перевірені усі числа переходимо по L2
        .END
```

Приклад 5

Написати програму на мові Асемблер для мікроконтролера AT90S2313, яка здійснює обмін змісту блоків пам'яті, що розташовані починаючи з адреси 0060H та 00A0H, які містять по 5 байт.

Для обміну змісту блоків пам’яті будуть задіяні дві регістрові пари Z, Y в які будуть завантаженні адреси комірок пам’яті при використовуванні непрямої адресації. Для підрахунку кількості чисел, які переписуються, використовується регістр-лічильник. З нього після виконання обміну даними між комірками буде відніматись 1. Коли зміст регістра-лічильника буде дорівнювати 0, це означає, що обмін змісту блоків пам’яті виконаний задане число раз.

```
        LDI R20, 05      ; запис до регістра-лічильник кількість байт для обміну
        CLR R31          ; завантаження да регістрової пари Z адреси 0060H
        LDI R30, $60
        CLR R29          ; завантаження до регістрової пару Y адреси 00A0H
        LDI R28, $A0
L1:     LD R2, Z         ; зчитати число за адресою 0060H (регістрова пара Z) до R2
        LD R3, Y         ;зчитати число за адресою 00A0H (регістрова пара Y) до R3
        ST Z, R3         ; записати число з R3до комірки адреса якої знаходиться в Z
        ST Y, R2         ; записати число з R2 до комірки адреса якої знаходиться в Y
        INC R28          ; перейти до наступної комірки пам’яті, адреса якої в Y
        INC R30          ; перейти до наступної комірки пам’яті, адреса якої в Z
        DEC R20          ; відняти 1 від регістра-лічильника
        TST R20          ; перевірити чи регістр-лічильник не дорівнює 0
        BRNE L1          ; якщо ні, то перейти за міткою L1
        NOP              ; кінець виконання програми
```

Приклад 6

Розробити програму на мові Асемблер для мікроконтролера AT90S2313, яка пересилає 10 чисел з комірки SRAM–80h в комірку SRAM–A0h.

```
        CLR R31
        LDI R30,$80      ;завантаження до рег. пари Z адреси комірки SRAM–80h
        CLR R29
        LDI R28,$A0      ;завантаження до рег. пари Y адреси комірки SRAM–A0h
        LDI R16, 10      ;лічильник
L2:     LD R5, Z+        ;зчитати до R5 перше число та збільшити адресу, що знаходиться
                         ;в Z на 1
        ST Y+, R5        ;записати з R5 перше число та збільшити адресу, що знаходиться в
                         ;Y  на 1
        DEC R16          ;відняти 1 від лічильника чисел
        BRNE L2          ;якщо R16 не дорівнює 0, то повторити. Для цього перейти по
                         ;мітці  L2:
        NOP
```

Приклад 7

Розробити програму на мові Асемблер для мікроконтролера AT90S2313, яка логічно множить 8 чисел, що знаходяться в комірках SRAM з адресою

70h. Результат зберегти в регістрі R1

```
        CLR R31         ;очистити старшу частину регістрової пари Z
        LDI R30, $70    ;запис у молодшу частину регістрової  пари Z адресу $70
        LDI R20, 8      ;запис у регістр (лічильник) кількості чисел
        LDI R16, $FF    ;запис до R16 числа   $FF
        MOV R1, R16     ;запис в R1  числа   $FF
L1:     LD R16, z+      ;зчитати перше число за адресою $0070 до R16 та збільшити адресу
                        ;на 1
        AND R1, R16     ;логічно перемножити R1, R16. Результат в  R1.
        DEC R20         ;зменшити кількість чисел на 1
        BRNE L1         ;якщо не всі числа логічно перемножили, то перейти по мітці  L1
        NOP             ;кінець
```

Приклад 8

Зчитати байт з комірки адреса якої в Z, додати до нього $08. Результат помістити в комірку з адресою в Y.

```
LPM             ;зчитати вміст регістрової пари Z в регістр R0
LDI R17, $08    ;записати в R17 константу $08
ADD R0, R17     ;додати до числа в R0 число в R17, результат залишається в R0
ST Y, R0        ;зберегти вміст регістра R0 в регістровій парі Y
```

Приклад 9

Перевірити чи в SRAM - $AA 7 біт дорівнює 1. Якщо так то зміст комірки $A0 зменшити на $05, якщо ні то збільшити на $05. Результат переслати в регістр R28.

Варіант а

```
        LDS R28, $A0    ;записати в R28 вміст комірки SRAM з адресою $A0
        LDS R5, $AA     ;записати в R5 вміст комірки SRAM з адресою $AA
        LDI R16, $05    ;завантажити в R16 константу $05
        BST R5, 7       ;зберегти 7 біт з регістра R5 в T
        BRTS L1:        ;перейти якщо прапор T встановлений
        ADD R28, R16    ;додати до числа в R28 число в R16, результат залишається в R28
L1:     SUB R28, R16    ; R28 –R16,  результат залишається в R28
```

Варіант б

```
        LDS R28, $A0    ;записати в R28 вміст комірки SRAM з адресою $A0
        LDS R25, $AA    ;записати в R25 вміст комірки SRAM з адресою $AA
        LDI R16, $05    ;завантажити в R16 константу $05
        ANDI R25, 80h   ;маскування
        TST R25         ;перевірка на нуль вмісту R5
        BREQ L1:        ;перейти якщо вміст R5 рівний 0
        SUB R28, R16    ; R28 –R16,  результат залишається в R28
```

```
L1:  ADD R28, R16   ;додати до числа в R28 число в R6, результат залишається в R28
                                                                          Варіант в
     LDS R28, $A0   ;записати в R28 вміст комірки SRAM з адресою $A0
     LDS R5, $AA    ;записати в R5 вміст комірки SRAM з адресою $AA
     LDI R16, $05   ;завантажити в R16 константу $05
     SBRC R5, 7     ;пропустити наступну команду якщо 7-й біт рівний 0
     SUB R28, R16   ; R28 –R16,  результат залишається в R28
     ADD R28, R16   ;додати до числа в R28 число в R16, результат залишається в R28
```

Приклад 10

Прочитати з комірки SRAM пам'яті $65 байт і якщо він рівний 55h збільшити зміст R2 на 1, в противному випадку зменшити на 1.

```
     LDS R25, $0065   ;завантажити число з комірки за адресою $0065      варіант а
     CPI R25, $55     ;порівняти вміст R25 з константою $55
     BREQ L1:         ;перейти до мітки L1: якщо рівні значення
     DEC R2           ;зменшити вміст R2 на 1
     RJMP L2          ;перейти до кінця програми
L1:  INC R2           ;збільшити вміст R2 на 1
L2:  NOP
```

```
     CLR R31          ;записати в регістрову пару Z адресу $65(перші       варіант б
     LDI R30, $65     ;дві команди це ініціалізація регістрової пари Z)
     LD R20, Z        ;завантажити в R20 вміст комірки з адресою $65
     SUBI R20, $55    ;відняти від вмісту R20 константу $55
     TST R20          ;перевірка R20 на нуль
     BREQ L1:         ;перейти до мітки L1: якщо рівні значення
     DEC R2           ;зменшити вміст R2 на 1
     RJMP L2          ;перейти до кінця програми
L1:  INC R2           ;збільшити вміст R2 на 1
L2:  NOP
```

```
     CLR R31          ;очистити регістр R31                               варіант в
     LDI R30, $65     ;записати в регістрову пару Z адресу $65
     LD R20, Z        ;завантажити в R20 вміст комірки з адресою $65
     BRNE L1:         ;перейти якщо не дорівнює
     INC R2           ;збільшити вміст R2 на 1
     RJMP L2          ;перейти до кінця програми
L1:  DEC R26          ;зменшити вміст R2 на 1
L2:  NOP
```

Приклад 11

Якщо z=1 то перейти до виконання підпрограми, якщо z=0 то $60…$80

знайти код $0A і адресу байту помістити до стеку.

```
        SBIC $3F, 1     ;пропустити якщо біт очищений
        RCALL  PP       ;виклик підпрограми
        CLR R31         ;очистити регістр R31
        LDI R30, $60    ;записати в регістрову пару Z адресу $DF
L2:     LD R20, X       ;зчитати вміст регістрової пари Z в регістр R20
        CPI R20, $0A    ;порівняти вміст R20 з константою $0A
        BREQ L1:        ;перейти якщо вміст R0 рівний $0A
        INC R30         ;збільшити вміст R30 на 1
        CPI R30, $80    ;порівняти вміст R30 з константою $80
        BRNE L2:        ;перейти якщо вміст R30 не дорівнює $80
L1:     PUSH R30        ;занести вміст R30 в стек
```

Приклад 12

Зчитати дані з EEPROM по адресу $70 в регістр R6

```
LDI R20, $70        ;записати в R20 константу $70
OUT $1E, R20        ;записати адресу в EEPROM
SBI $1C, 0          ;встановити біт читання а EECR
IN R6, $1D          ;зчитати число з EEPROM
```

Приклад 13

Знайти найменше число масиву $60...$80 і записати його в R2. Якщо воно менше $25 то записати його в SRAM по адресу $A0, якщо воно більше – в $A1

```
        CLR R27         ;очистити регістр R27
        LDI R26, $60    ;записати в регістрову пару X адресу $60 (ініціалізація рег. пари X)
        LD R0, X+       ;завантажити в R0 число з комірки адреса якої знаходиться в
                        ;регістровій парі X, з подальшим збільшенням змісту X на 1
        MOV R22, R0     ;скопіювати зміст R0 в R2
L2:     LD R0, X+       ;завантажити в R0 число з комірки адреса якої знаходиться в
                        ;регістровій парі X, з подальшим збільшенням змісту X на 1
        CP R0, R22      ;порівняти зміст регістрів
        BRGE L1:        ;якщо R0>R2 то перейти до мітки L1:
        MOV R22, R0     ;скопіювати зміст R0 в R22
L1:     CPI R26, $80    ;порівняти вміст R26 з константою $80
        BRNE L2:        ;якщо не ріні значення перейти до мітки L2
        CPI R22, $25    ;порівняти вміст R2 з константою $25
        BRLO L3:        ;якщо менше перейти до мітки L3
        STS $A1, R22    ;зберегти вміст R22 в комірці SRAM з адресою $A1
        RJMP STP:       ;перехід до мітки STP
```

```
L3:   STS $A0, R22   ;зберегти вміст R22 в комірці SRAM з адресою $A0
STP:  NOP            ;немає операції
```

Приклад 14

Знайти найбільше число масиву чисел SRAM $68-$8F.

```
      CLR R27        ;очистити регістр R27
      LDI R26, $68   ;записати в регістрову пару X адресу $68 (ініціалізація рег. пари X)
      LD R0, X+      ;завантажити в R0 число з комірки адреса якої знаходиться в
                     ;регістровій парі X, з подальшим збільшенням змісту X на 1
      MOV R22, R0    ;скопіювати зміст R0 в R22
L2:   LD R0, X+      ;завантажити в R0 число з комірки адреса якої знаходиться в
                     ;регістровій парі X, з подальшим збільшенням змісту X на 1
      CP R0, R22     ;порівняти зміст регістрів
      BRLO L1:       ;якщо R22>R0 то перейти до мітки L1:
      MOV R22, R0    ;скопіювати зміст R0 в R22
L1:   CPI R26, $8F   ;порівняти вміст R26 з константою $8F
      BRNE L2:       ;якщо не рівні значення перейти до мітки L2
      NOP            ;немає операції
```

Приклад 15

Знайти функцію y = (x_1+x_2)$08, результат зберегти в R20, де x_1 – число, яке зчитується з порта B, x2 – число, яке знаходиться в регістрі R6.*

```
CLR R16        ;очистити R16
OUT $17, R16   ;ініціалізація порта B на ввід
IN R22, $16    ;зчитати значення з порта B
OR R22, R16    ;логічне АБО між R22 і R16
ANDI R22, $08  ;логічне І між R22 і константою
MOV R20, R22   ;скопіювати R22 в R20
```

Приклад 16

Якщо число в порту B перевищує $28, то видати сигнал тривоги через порт D

```
      LDI R16, $FF     ; до R16 заносимо число 11111111b
      OUT DDRD, R16    ; порт D настроєний на ввід інформації
      LDI R16, $00     ; до R16 заносимо число 000000000b
      OUT DDRB, R16    ; порт B настроєний на ввід інформації
L3:   IN R20, PinB     ; зчитати число з порта B в R20
      CPI R20, $28     ; порівняти R20 з $28
      BRLO L1          ; якщо число менше $28. То перейти по L1, інакше виконати
                       ; наступну команду
      SBI PORTD, 1     ; видати сигнал тривоги через 1 біт порта D
      RJMP L3          ; перейти по L3
```

```
L1:     CBI PORTD, 1        ; відключити  сигнал тривоги
L2:     Rjmp L3             ; перейти по L3
```

Приклад 17

До порта B підключені 8 світлодіодів. Записати в кодах AT90S2313 програму, яка відображає на індикаторі по черзі молодшу та старшу частину числа з регістру R26.

```
        LDI R20, $FF        ; запис в регістр R20 числа $FF
        OUT DDRB, R20       ; настроїти порт B на вивід інформації
        MOV R21, R26        ; скопіювати зміст R26 в R21
        ANDI R21, $0F       ; виділення молодшої частини числа
L1:     OUT PORTB, R21      ; вивід молодшої частини числа в порт B для відображення на
                            ; світлодіодах
        RCALL DELAY         ; виклик підпрограми затримки
        ANDI R26, $F0       ; виділення старшої частини числа
        OUT PORTB, R26      ; вивід старшої частини числа в порт B для відображення на
                            ; світлодіодах
        RCALL DELAY         ; виклик підпрограми затримки
        RJMP L1             ; повторення виводу молодшої та старшої частини числа
```

Приклад 17

Розробити програму, яка створює ефект «бігучі вогники» на мові Асемблер для мікроконтролера AT90S2313, якщо до порту B підключено 8 світодіодів.

```
        LDI R20, $FF        ; запис в регістр R20 числа $FF
        OUT DDRB , R20      ; настроїти порт B на вивід інформації
        LDI R20, $FE        ; запис в R20 числа b11111110 (світиться молодший світлодіод)
        SEC                 ; установити флаг переносу
L1:     OUT PORTB, R20      ; вивести світловий ефект
        RCALL DELAY         ; виклик підпрограми затримки
        ROL R20             ; зсув ліворуч через перенос
        RJMP L1             ; повторення програми
```

Приклад 18

Розробити програму, яка створює ефект «миготіння» на мові Асемблер для мікроконтролера AT90S2313, якщо до порту B підключено 8 світодіодів.

```
        LDI R20, $FF        ; запис в регістр R20 числа $FF
        OUT DDRB , R20      ; настроїти порт B на вивід інформації
        CLR R21             ; скопіювати зміст R26 в R21
L1:     OUT PORTB, R20      ; усі світлодіоди світяться
```

```
        RCALL DELAY     ; виклик підпрограми затримки
        OUT PORTB, R21  ; усі світлодіоди погашені
        RCALL DELAY     ; виклик підпрограми затримки
        RJMP L1         ; повторення програми
```

Приклад 19

Розробити програму формування прямокутних імпульсів з періодом 100мкс на мові Асемблер для мікроконтролера AT90S2313, якщо тактова частота мікро контролера 1МГц.

```
        LDI R16, RamEnd     ; до R16 заносимо адресу останньої комірки SRAM
        OUT SPL, R16        ; налаштування вершини стеку
        LDI R16, $FF        ; до R16 заносимо число 11111111b
        OUT PORTB, R16      ; порт B настроєний на вивід інформації
L1:     SBI PORTB, 0        ; установити 1  на 0 виводі порта B
        RCALL DEL           ; виклик підпрограми затримки на 50 мкс
        CBI PORTB, 0        ; установити 0  на 0 виводі порта B
        RCALL DEL           ; виклик підпрограми затримки на 50 мкс
        RJMP L1             ; повторення програми формування прямокутного імпульса
DEL:    LDI R20, 25         ; п/п затримки. Формує затримку 50мкс, для того щоб період
L5:     DEC R20             ; імпульсів був 100 мкс
        BRNE L5             ;
        RET                 ;
```

Приклад 20

*Виконати функцію Y=x1+x2*x3 – x4, де x1 – число, яке необхідно зчитати з порта B, x2 – з порта D, x3 – з EEPROM з адресою $10, x4 – число, яке знаходиться в комірці пам'яті $68.*

```
CLR R20         ; очистити регістр R20
OUT DDRB, R20   ; настроїти порт B на ввід інформації
OUT DDRD, R20   ; настроїти порт B на ввід інформації
IN R16, PinB    ; помістити перше число в R16 (зчитується з порта B)
IN R17, PinD    ; помістити друге число в R17 (зчитується з порта D)
LDI R20, $10    ; запис в R20 числа $10
OUT EEAR, R20   ; у регістр адреси EEPROM записується адреса з R20
SBI EECR, EERE  ; у регістрі керування EEPROM встановлюється біт дозволу читання
IN R18, EEDR    ; помістити третє число в R18 (зчитується з EEPROM)
LDS R19, $68    ; помістити четверте число в R19 з комірки пам'яті  з адресою $68
ADD R16, R17    ;x1+x2
AND R16, R18    ;x1+x2*x3
SUB R16, R19    ;x1+x2*x3 – x4
```

Перелік задач до самостійного розв'язку

1. В області SRAM $60-$80 визначити мінімальне число і записати його в R2. Якщо воно менше $3F, то записати його в SRAM з адресою $AF, якщо більше то записати його в SRAM з адресою $AE.
2. В області SRAM $60-$DF перевірити чи є запис числа $0A, якщо вони є замінити їх на $0F, а в регістр R18 записати кількість замінених байтів.
3. Викликати безперервні спалахи світлодіода з інтервалом 1с, який підключений до 3 розряду порта B.
4. Виконати функцію Y=x1+x2*x3 – x4, де x1 – число, яке необхідно зчитати з порта D, x2 – з порта B, x3 – число, яке знаходиться в комірці пам'яті $C8, x4 – з EEPROM з адресою $18.
5. Відобразити на індикаторі число 21. По натисканню кнопки S1 зменшувати на одиницю відображаємо число, по натисканню S2 збільшувати на одиницю відображаємо число.
6. Відобразити на індикаторі число 8. По натисканню кнопки S9 зменшувати на одиницю, відображаєме число, по натисканню S8 збільшувати на одиницю відображаємо число
7. Відсортувати масив чисел SRAM $90-$CF в порядку зростання.
8. До порта D підключені 7 світлодіодів. Записати в кодах AT90S2313 програму, яка відображає на індикаторі по черзі молодшу та старшу частину числа з регістру R26.
9. Знайти найбільше та найменше число масиву чисел SRAM $80-$9F.
10. Знайти найбільше число масиву чисел SRAM $68-$8F.
11. Знайти найменше число масиву чисел SRAM $60-$DF.
12. Знайти функцію Y=(x1+x2)*$08, де x1 - число, яке необхідно зчитати з port B, x2 – число, яке необхідно зчитати з EEPROM ($32). Результат помістити в R20.
13. Зчитати 20 чисел з 12 розрядного АЦП. Результат зберегти в комірках пам'яті починаючи з адреси 80h. Розробити функціональну схему та написати програму на мові Асемблер.
14. Зчитати 30 чисел з 14 розрядного АЦП. Результат зберегти в комірках пам'яті починаючи з адреси 82h.Розробити функціональну схему та написати програму на мові Асемблер
15. Зчитати 30 чисел з 8 розрядного АЦП, всі числа що перевищують 100 переписати в комірки пам'яті SRAM починаючи з $60, всі числа що менше 10 переписати в комірки пам'яті SRAM починаючи з $A0. Додати всі числа, результат зберегти в R8.
16. Зчитати дані з EEPROM (адреса $30) в комірку пам'яті SRAM $DF, та з комірки SRAM $60 записати число в EEPROM.
17. Зчитати число з $A1 й порівняти його з $55. Якщо число більше за $55 то записати його в комірку на 1 менше вихідної, якщо менше – то в комірку на 1 більше.
18. Зчитати число з порта B в R6 й порівняти його з $55, якщо число більше то записати його в стек, якщо ні то записати його в R3.
19. Необхідно вивести 20 даних з ОЗП з адресою $60 на ЦАП, прочитати дані з АЦП , записати зчитані дані в ОЗП по адресу, починаючи з $60
20. Нехай у пам'яті програм, починаючи з комірки $68 SRAM, розташована таблиця кодів довжиною (X1, i = 1,2,...…....D, формат – байт). Записати в кодах AT90S2313 програму, що виконує обчислення заданої функції Min (Xi)\/Xn над цими кодами. Результат обчислення розмістити в комірку з $99.
21. Нехай у пам'яті програм, починаючи з комірки $88 SRAM, розташована таблиця кодів довжиною (X1, i = 1,2,...…....C, формат – байт). Записати в кодах AT90S2313 програму, що виконує обчислення заданої функції Max(Xi)\/Xn над цими кодами. Результат обчислення розмістити в комірку з $90.

22. Перевірити, чи в SRAM $AA 7 біт дорівнює 1, якщо так то зміст комірки пам'яті $A0 зменшити на $05, якщо ні то збільшити на $05. Результат переслати в регістр R28.
23. У двійковому виді відображати на одиничних індикаторах номер натиснутої кнопки.
24. Якщо натиснута кнопка від 0 до 4 то відобразити на індикаторі номер цієї кнопки, в усіх інших випадках горять всі індикатори.
25. Якщо ознака Z=1, то перейти на виконання підпрограми, якщо Z=0 то з $60-$80 знайти код $0A і адресу байту помістити до стеку.
26. Якщо число в порту B перевищує $28, то видати сигнал тривоги через порт D.
27. Розробити програму формування прямокутних імпульсів з періодом 100мкс на мові Асемблер для мікроконтролера AT90S2313, якщо тактова частота мікро контролера 1МГц.
28. Розробити програму формування трикутного імпульсу з максимальною амплітудою на мові Асемблер для мікроконтролера AT90S2313, якщо до порту B підключений 8 бітний ЦАП.
29. Розробити програму формування пилкоподібного імпульсу з максимальною амплітудою на мові Асемблер для мікроконтролера AT90S2313, якщо до порту B підключений 8 бітний ЦАП.
30. Розробити програму, яка створює ефект «бігучі вогники» на мові Асемблер для мікроконтролера AT90S2313, якщо до порту B підключено 8 світодіодів.
31. Розробити програму, яка створює ефект «бігуча тінь» на мові Асемблер для мікроконтролера AT90S2313, якщо до порту B підключено 8 світодіодів.
32. Розробити програму, яка створює ефект «миготіння» на мові Асемблер для мікроконтролера AT90S2313, якщо до порту B підключено 8 світодіодів.
33. Розробити програму, яка перевіряє стан кнопки S1 на мові Асемблер для мікроконтролера AT90S2313, якщо до порту B підключено 8 світодіодів. Якщо кнопка не натиснута (S1=1) на індикаторі виводиться число 0Fh, якщо натиснута S1=0 то виводиться число FFh. Кнопка S1 підключена до PD0.
34. Розробити програму на мові Асемблер для мікроконтролера AT90S2313, яка пересилає 10 чисел з комірки SRAM–80h в комірку SRAM–A0h.
35. Розробити програму на мові Асемблер для мікроконтролера AT90S2313, яка пересилає 10 чисел, що зчитуються з порта B в комірки SRAM починаючи з адреси 80h.
36. Розробити програму на мові Асемблер для мікроконтролера AT90S2313, яка додає 10 чисел, що зчитуються з порта B. Результат записати в комірку SRAM з адресою 72h.
37. Розробити програму на мові Асемблер для мікроконтролера AT90S2313, яка додає 12 чисел, що знаходяться в комірках SRAM з адресою 60h. Результат зберегти в регістрі R20.
38. Розробити програму на мові Асемблер для мікроконтролера AT90S2313, яка логічно додає 15 чисел, що знаходяться в комірках SRAM з адресою 90h. Результат зберегти в регістрі R10.
39. Розробити програму на мові Асемблер для мікроконтролера AT90S2313, яка логічно множить 8 чисел, що знаходяться в комірках SRAM з адресою 70h. Результат зберегти в регістрі R1
40. Написати програму на мові Асемблер для мікроконтролера AT90S2313 для перевірки працездатності комірок SRAM з адреси 60h по адресу 80h. Для перевірки використовується наступний алгоритм: в комірку записати всі одиниці, зчитати та перевірити правильність. Якщо є помилка то повідомити світінням світлодіода (PD0), та вийти з режиму перевірки. Кінець перевірки повідомляється світінням світлодіода (PD1).