# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»**

**Лабораторні роботи з курсу**

**Проектування мікропроцесорних систем**

*Виконав*

студент групи ЗПІ-ЗП 61

(Варіант №2)

Макіян Смбат Артурович

*Перевірив*

Викладач

Долина В.Г.

# Київ 2019

# Лабораторна робота №1

**Завдання А:** Вміст UDR помістити в пам’ять даних.

*Алгоритм*: Скопіювати значення UDR в R16, записати значення R16 в пам'ять даних.

*Лістинг алгоритму:*

|  |
| --- |
| .include "8515def.inc";Includes the 8515 definitions file  in R16, UDR  sts $60, R16 |

**Завдання В:** Організувати під стек пам'ять даних з адреси 70Н і помістити в стек вміст таймера/лічильника1 і порта D (PIND).

*Алгоритм:* Для початку ініціалізуєм R16 константою 70h, після цього занесемо SPL. Після цього зчитуємо PIND в R16 та занесемо її до стеку командой PUSH.

*Лістинг алгоритму:*

|  |
| --- |
| .include "8515def.inc" ;Includes the 8515 definitions file  ; init stack 70h  LDI R16, 0x70  OUT SPL, R16  ; read PIND  in R16, PIND  ; write to stack  PUSH R16 |

**Завдання С:** За адресою 60H в пам'яті даних знаходиться таблиця 3х5, перші 3 елемента останньої строки помістити в пам'ять даних, починаючи з адреси 80Н.

*Алгоритм*: Для початку зчитуємо значення за адресою 0x6A до R16 командою lds після цього записуємо його до 0x80. Те

*Лістинг алгоритму:*

|  |
| --- |
| .include "8515def.inc ;Includes the 8515 definitions file  EOR R16, R16  lds R16, 0x6A  sts R16, 0x80  lds R16, 0x6B  sts R16, 0x81  lds R16, 0x6C  sts R16, 0x82 |

# Лабораторна робота №2

**Завдання А:** Обчислити суму від'ємних чисел від -1 до -10.

*Алгоритм*: Для додавання даних використовуємо декілька регістрів: R18 – константа порівняння для виходу з цикла; R19 – константа декременту; R16 реєстр в котрому буде зберігатись результат обчислення, R17 – реэстр котрий будемо додавати.

*Лістинг алгоритму:*

|  |
| --- |
| .include "8515def.inc ;Includes the 8515 definitions file  ; sum of negative -1 -> -10  LDI R16, -1  LDI R17, -2  LDI R18, -10  LDI R19, -1  FLOOP:  ADD R16, R17  SUB R17, R19  CPSE R17, R18  RJMP FLOOP |

**Завдання В:** Просумувати два числа, що розташовані за адресою 7Dh і 7Eh, результат помістити в регістр R29.

*Алгоритм*: Зчитуємо 0x7D до R17, зчитуємо 0x7E до R18, додаємо R17 до R29 командою add, повторюємо теж саме для R18.

*Лістинг алгоритму:*

|  |
| --- |
| .include "8515def.inc ;Includes the 8515 definitions file  ; add 0x7Dh and 0x7eh  EOR R17, R17  EOR R18, R18  LDS R17, 0X7D  LDS R18, 0X7E  EOR R29, R29  ADD R29, R17  ADD R29, R18 |

**Завдання С:** Знайти суму діагональних (головної і додаткової діагоналей) елементів матриці 2\*2, що розташована починаючи з адреси 60h.

*Алгоритм*: У регістри R16 та R17 головної діагоналі 0x60, 0x63, занулім R19, та додамо ці регістри до нього. Після цього зчитуємо до регістрів R16 та R17 елементи побічної діагоналі, занулім регістр R20 та додамо ці регістри до нього.

*Лістинг алгоритму:*

|  |
| --- |
| .include "8515def.inc ;Includes the 8515 definitions file  EOR R16, R16  EOR R17, R17  EOR R19, R19  EOR R20, R20  LDS R16, 0x60  LDS R17, 0x63  ADD R19, R16  ADD R19, R20  LDS R16, 0x61  LDS R17, 0x62  ADD R20, R16  ADD R20, R16 |

# Лабораторна робота №3

**Завдання А:** Скинути перший біт регістра керування таймера\лічильника1 (TCCR1B).

*Алгоритм*: TCCR1B являє собою регістр керування таймера\лічильника1 та містить 8 бітів. Перші три біти мають назву CS10(0), CS11(1), CS12(2) та відповідають за вибір тактової частоти (коефіцієнту ділення).

*Лістинг алгоритму:*

|  |
| --- |
| .include "8515def.inc" ; Includes the 8515 definitions file  ldi r16, (0<<CS10) ;зберігаємо данні про стан першого біту (скидаємо на 0)  out tccr1b, r16 ;виконуємо застосування змін |

**Завдання В:** Реалізувати комбінаційну двійкову функцію, задану аналітично:

bit1 =

*Алгоритм*: Для підрахунку результату (bit1) необхідно виконати додавання 4 біту регістру R20 з 3 бітом регістру R21, та виконати команду neg. Для додавання регістрів необхідно виконати зсув регістру R20 вправо, щоб 4 біт став 3-ім. Після зсуву виконується додавання та інверсія бітів. Результат буде знаходитися у 3 біті регістру R21, томе необхідно буде виконати зсув до 3-го біту.

*Лістинг алгоритму:*

|  |
| --- |
| .include "8515def.inc" ; Includes the 8515 definitions file  lsr R20 ;Виконуємо зсув, щоб 4-й біт став 3-ім у регістрі 20  add R21,R20 ;Додаємо регістри  neg R21 ;Інвертуємо регістр  lsr R21 ;Виконуємо зсув для отримання результату у біті №3  lsr R21  lsr R21 ;4-й біт став першим, в ньому наш результат |

**Завдання С:** Реалізувати функцію задану у таблиці:



*Алгоритм*: Проаналізувавши функцію можна зробити висновок та розклаши її за картами Карно, то бачимо, що F дорівнює bit1.

*Лістинг алгоритму:*

|  |
| --- |
| .include "8515def.inc" ; Includes the 8515 definitions file  EOR R15, R15  EOR R16, R16 ; this is should be our result  LDS R15, 0x01 ; load constant  MOV R16, R15 |

# Лабораторна робота №4

**Завдання А:** У виконуваному коді є 4 підпрограми. Початкові адреси дані в 4-х парах регістрів R11 і R12, R13 і R14, .... В залежності від вмісту R30 - дорівнює 1, 2, 3 або 4 - перейти до виконання однієї з підпрограм.

*Алгоритм*: Для виконання переходу за умови (case) записуємо варіанти умов у регістри R26, R27, R28, R29. Наступним кроком виконуємо порівняння значення регістру R30 з іншим регістром, командою cp та виконуємо перехід, якщо значення рівні, командою breq. Після виконання переходу, записуємо початкове значення, що надане у одному із регістрів R11, R12, R13 R14 у регістр R31. Регістр R31 містить початкову адресу однієї із мікропрограм.

*Лістинг алгоритму:*

|  |
| --- |
| .include "8515def.inc" ; Includes the 8515 definitions file  LDI R26,$1 ; filling jump condition  LDI R27,$2  LDI R28,$3  LDI R29,$4  LDI R30,$3 ; jump check  CP R30,R26 ; compare 2 values  BREQ JMP1 ; jump if exuals  CP R30,R27  BREQ JMP2  CP R30,R28  BREQ JMP3  CP R30,R29  BREQ JMP4  JMP1: MOV R31,R11 ; write first address to R31  JMP2: MOV R31,R12  JMP3: MOV R31,R13  JMP4: MOV R31,R14  PUSH R31 ; Adding R31 to stack |

**Завдання В:** У регістрах R11 і R12 містяться числа. Якщо при складанні цих чисел прапорець переповнення встановиться в 1, то необхідно в регістр R17 записати число 21H, в іншому випадку в регістр R18 записати число 31H h

*Алгоритм*: Першим кроком необхідно виконати додавання двох регістрів R11 і R12. Для переходу по переповненню використовуємо команду brvs, яка спрацьовує, якщо флаг переповнення V буде дорівнювати 1. Якщо флаг спрацював, виконуємо запис в регістр R17 числа 21H, якщо ні - в регістр R18 записати число 31H.

*Лістинг алгоритму:*

|  |
| --- |
| .include "8515def.inc" ; Includes the 8515 definitions file  LDI R26,$35 ; SET INITIAL DATA  LDI R27,$5B  MOV R11,R26 ;ADD DATA TO REGISTERS  MOV R12,R27  ADD R11,R12 : ADD  BRVS OVERFL ; jmp is overflow  LDI R18,$31 ; statement 2 if ok  RET  OVERFL: ; in case of overflow  LDI R17,$21  RJMP EXIT2  EXIT2:  RET |