Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

**Отчёт по лабораторной работе №5**

«Калькулятор»

по дисциплине "Микропроцессорные системы»

Работу выполнил студент

Дорофеев Юрий Владимирович гр. 4081/12 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работу принял преподаватель

Павловский Евгений Григорьевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Санкт-Петербург

2013

Содержание отчета

[1. Цель работы 3](#_Toc349609362)

[2. Программа работы 3](#_Toc349609363)

[3. Теоретические положения 3](#_Toc349609364)

[4. Алгоритм работы 7](#_Toc349609365)

[5. Работа с программой 9](#_Toc349609366)

[6. Текст программы 10](#_Toc349609367)

[7. Выводы 23](#_Toc349609370)

1. Цель работы

Проверить индивидуальные знания студентов. Изучить и применить на практике основы построения специализированных вычислителей на примере микроконтроллеров. Обобщить знания по курсу «Микро ЭВМ».

1. Программа работы

Реализовать 2-х байтовый знаковый калькулятор с поддержкой клавиатуры, ЖКИ и простых арифметических операций.

1. Теоретические положения

2-х байтовые операции реализуются следующим образом:

**Сложение:**

*Bh,Bl + Ah,Al*

Реализуется путем 2-х однобайтовых сложений *Bl+Al* без учета переноса и *Bh + Ah* с учетом переноса (если выставлен бит *c*, то переполнение).

**Вычитание:**

*Bh,Bl – Ah,Al*

Реализуется путем 2-х однобайтовых вычитаний *Bl-Al* и *Bh-Ah* с учетом займа (если выставлен бит *c*, то отрицательный результат).

**Умножение:**

Реализуется тремя 1- байтовыми умножениями и сложением.

*Bh,Bl*\**Ah,Al*=*Ch,Cl*

* *Cl*=*Bl*\**Al;*
* *Ch*=*Bh*\**Al*+*Ah*\**Bl*.

Если *Bh*\**Ah* не равно 0, то переполнение.

**Деление:**

Реализуется известным алгоритмом деления с восстановлением остатка, представленным на рис. 3.

**Схема устройства:**



*Рис.1. Схема устройства*



**Граф состояний программы и алгоритм работы:**

*Рис.2. Граф состояний*

**Алгоритм деления:**

*Рис 3 Алгоритм деления*

Знак хранится в старшем разряде старшего бита.

Знаковые операции в 2-х байтовом калькуляторе мы реализовали следующим образом:

**Умножение:**

*ЗнС*=*ЗнА* xor *ЗнB*

**Деление:**

ЗнС=ЗнА xor ЗнB

**Архитектура:**

При описании реализации калькулятора на базе МК используются понятия регистров A, B. Физически это 2-х байтовые переменные в памяти МК.

1. Алгоритм работы

**Шаг 1**. Если нажата клавиша, определяется ее код, иначе переходим к шагу 1

**Шаг 2**. Если код – цифра то переходим к Шагу 3, иначе к шагу 4

**Шаг 3**. Умножаем содержимое регистра А на 10 (А – 2 байта), прибавляем к содержимому А полученную цифру. Выводим А на ЖКИ. Переход к Шагу 1

**Шаг 4**. Если код функциональная клавиша, то Флаг функциональной клавиши = 1 и переход к Шагу 1

**Шаг 5**. Если нажатая клавиша соответствует операции (+,-,\*,/), то в регистр А записываем содержимое регистра B, В регистр операций записываем код операции, переходим к Шагу 1, иначе к Шагу 6

**Шаг 6**. Если нажат унарный минус, то в А записываем –A, переходим к шагу 1, иначе к Шагу 7

**Шаг 7**. Если нажата клавиша «равно», то выполняем операцию в соответствии с кодом операции и результат помещаем в A и выводим на ЖКИ. Переход к Шагу 1

**Расположение клавиш на клавиатуре:**

*Таблица 1. Расположение клавиш на клавиатуре*

|  |  |  |  |
| --- | --- | --- | --- |
| **3** | **2** | **1** | **0** |
| **7** | **6** | **5** | **4** |
| **=** | **Унарный минус** | **9** | **8** |
| **-** | **+** | **/** | **\*** |

Так, как мы не можем работать с числами, занимающими больше чем два байта, введем искусственное ограничение на вводимые данные. Ограничим операнды тремя разрядами, т.е. 0-999. Для этого будем считать количество введенных символов (листинг 1).

Полный код программы представлен в листингах 2,3,4.

*Листинг 1. Ограничение на вводимые числа*

check: mov a, count

cjne a, #3, go\_norm

mov dptr, #data

mov a, '!'

movx @dptr, a

lcall indic

; Обнуление при неверном вводе

mov Al,#0

mov Ahi,#0

mov Chi,#0

mov Cl,#0

mov Bhi,#0

mov Bl,#0

lcall indicateA

mov Func,#0

mov LastKey,#20

mov OVF, #0

lcall loop

go\_norm:

ret

1. Работа с программой

* Вначале работы программы мы видим на ЖКИ пять нулей. Далее следует выполнить следующую последовательность действий:
* Нажать на кнопку «Унарный минус» (необязательный пункт»;
* Ввести первый операнд (введенные цифры выводиться на ЖКИ, в случае превышения трех разрядов выведется символ «!» и следует повторить ввод заново»;
* Ввести символ операции (отображается на ЖКИ);
* Ввести второй операнд;
* Нажать на клавишу «=»;
* Результат будет выведен на ЖКИ.

**Примеры работы программы**:

-100-(200)=300

100\*150=15000

-30/7=-4

1. Текст программы

*Листинг 2. Код файла main.asm*

Al: equ 41h

Ahi: equ 42h

Bl: equ 43h

Bhi: equ 44h

Chi: equ 48h

Cl: equ 49h

Dhi: equ 52h

Dl: equ 53h

Oper: equ 45h

Func: equ 46h

LastKey: equ 47h

OVF: equ 50h

cycle: equ 51h

razr\_num: equ 54h

signs: equ 70h

count: equ 71h

org 8100h

mov Al,#0

mov Ahi,#0

mov Chi,#0

mov Cl,#0

mov Bhi,#0

mov Bl,#0

lcall indicateA

mov Func,#0

mov LastKey,#20

mov OVF, #0

mov count, #0

loop: lcall klav

mov a, klav\_num

cjne a, LastKey, processK

ljmp loop

processK: mov LastKey, klav\_num

cjne a, #20, process

ljmp loop

process: lcall ProcessKey

ljmp loop

ProcessKey:

mov a, Func

cjne a, #1, NFKeyPress

lcall FuncKeyPress

ret

NFKeyPress:

mov a, klav\_num

cjne a, #11 , NoFunction

mov Func, #1

ret

NoFunction:

jc Number

cjne a, #12, Operation

;---------------------------

inc count

call check

;---------------------------

call Calc

ret

Number:

push Bl

push Bhi

mov Bhi, #0

mov Bl, #10

lcall Mult

mov a, OVF

cjne a, #1, next

pop Bhi

pop Bl

ret

next:

push Ahi

push Al

mov Ahi, Chi

mov Al, Cl

mov Bhi, #0

mov Bl, klav\_num

dec Bl

lcall Addd

mov a,OVF

cjne a, #1, next2

pop Al

pop Ahi

pop Bhi

pop Bl

ret

next2: mov Ahi, Chi

mov Al, Cl

pop a

pop a

pop Bhi

pop Bl

lcall indicateA

ret

Operation: mov Oper, klav\_num

mov Bhi, Ahi

mov Bl, Al

mov Ahi, #0

mov Al, #0

mov dptr, #data

mov a, dpl

clr c

add a, #20

mov dpl, a

mov a, dph

addc a, #0

mov dph, a

mov a, klav\_num

cjne a, #13, oper1

mov a, #42

movx @dptr, a

ret

oper1: cjne a, #14, oper2

mov a, #47

movx @dptr, a

ret

oper2: cjne a, #15, oper3

mov a, #43

movx @dptr, a

ret

oper3: mov a, #45

movx @dptr, a

ret

FuncKeyPress:

; ---------------------

mov count, #0

; -----------------------

mov Func, #0

mov a, klav\_num

cjne a, #14, FKP1

mov a, Ahi

xrl a, #128

mov Ahi, a

lcall indicateA

FKP1: mov Func, #0

ret

Calc: mov a, Oper

cjne a, #13, NotMul

lcall mul\_sign

mov a, OVF

cjne a, #0, CalcError

mov Ahi, Chi

mov Al, Cl

lcall indicateA

ret

NotMul: cjne a, #15, NotAdd

lcall add\_sign

mov a, OVF

cjne a, #0, CalcError

mov Ahi, Chi

mov Al, Cl

lcall indicateA

ret

NotAdd: cjne a, #16, NotSubb\_calc

lcall Subb\_calc

mov a, OVF

cjne a, #0, CalcError

mov Ahi, Chi

mov Al, Cl

lcall indicateA

ret

NotSubb\_calc:

cjne a, #14, NotDiv

lcall div\_sign

mov a, OVF

cjne a, #0, CalcError

mov Ahi, Chi

mov Al, Cl

lcall indicateA

NotDiv: ret

CalcError: mov dptr, #data

mov a, #101

movx @dptr, a

inc dptr

mov a, #114

movx @dptr,a

inc dptr

mov a, #114

movx @dptr, a

inc dptr

mov a, #111

movx @dptr, a

inc dptr

mov a, #114

movx @dptr, a

inc dptr

mov a, #33

movx @dptr, a

inc dptr

mov a, #33

movx @dptr, a

inc dptr

mov a, #33

movx @dptr, a

inc dptr

lcall indic

mov Ahi, #0

mov Al, #0

mov Bhi, #0

mov Bl, #0

ret

Mult: mov a, Al

mov b, Bl

mul ab

mov Chi, b

mov Cl, a

mov a, Ahi

mov b, Bl

mul ab

clr c

add a, Chi

mov Chi, a

jc Error

mov a, b

cjne a, #0, Error

mov a, Al

mov b, Bhi

mul ab

clr c

add a, Chi

mov Chi, a

jc Error

mov a, b

cjne a, #0, Error

mov a, Ahi

mov b, Bhi

mul ab

cjne a, #0, Error

mov a, b

cjne a, #0, Error

ret

Error: mov OVF, #1

ret

Addd: clr c

mov a, Al

add a, Bl

mov Cl, a

mov a, Ahi

addc a, Bhi

mov Chi, a

jnc AddRet

mov OVF, #1

AddRet: ret

mul\_sign: mov signs, #0

mov a, Ahi

anl a, #128

cjne a, #128, mul\_sign1

mov signs, #1

mul\_sign1:mov a, Ahi

anl a, #127

mov Ahi, a

mov a, Bhi

anl a, #128

cjne a, #128, mul\_sign2

mov a, signs

orl a, #2

mov signs, a

mul\_sign2: mov a, Bhi

anl a, #127

mov Bhi, a

lcall Mult

mov a, OVF

cjne a, #1, mul\_sign3

ret

mul\_sign3: mov a, Chi

anl a, #128

cjne a, #128, mul\_sign4

mov OVF, #1

ret

mul\_sign4: mov a, signs

cjne a, #0, mul\_sign5

ret

mul\_sign5: cjne a, #3, mul\_sign6

ret

mul\_sign6 mov a, Chi

orl a, #128

mov Chi, a

ret

div\_sign: mov signs, #0

mov a, Ahi

anl a, #128

cjne a, #128, div\_sign1

mov signs, #1

div\_sign1:mov a, Ahi

anl a, #127

mov Ahi, a

mov a, Bhi

anl a, #128

cjne a, #128, div\_sign2

mov a, signs

orl a, #2

mov signs, a

div\_sign2: mov a, Bhi

anl a, #127

mov Bhi, a

lcall divide

mov a, OVF

cjne a, #1, div\_sign3

ret

div\_sign3: mov a, Chi

anl a, #128

cjne a, #128, div\_sign4

mov OVF, #1

ret

div\_sign4: mov a, signs

cjne a, #0, div\_sign5

ret

div\_sign5: cjne a, #3, div\_sign6

ret

div\_sign6 mov a, Chi

orl a, #128

mov Chi, a

ret

divide: mov Chi,#0

mov Cl,#0

mov cycle, #0

mov razr\_num, #0

mov a, Bhi

cjne a, Ahi, divide\_cor

mov a, Bl

cjne a, Al, divide\_cor

mov Chi,#0

mov Cl, #1

ret

divide\_cor:

jnc normalisB

ret

;----------------

check: mov a, count

cjne a, #3, go\_norm

mov dptr, #data

mov a, '!'

movx @dptr, a

lcall indic

; initilizto ZERo cond

mov Al,#0

mov Ahi,#0

mov Chi,#0

mov Cl,#0

mov Bhi,#0

mov Bl,#0

lcall indicateA

mov Func,#0

mov LastKey,#20

mov OVF, #0

lcall loop

go\_norm:

ret

;----------------------

normalisB: mov a, Bhi

anl a, #128

cjne a, #0, normalis

inc cycle

mov a, Bl

clr c

rlc a

mov Bl, a

mov a, Bhi

rlc a

mov Bhi, a

ljmp normalisB

normalis: mov a, Ahi

anl a, #128

cjne a, #0, Norm\_d

mov a, Al

clr c

rlc a

mov Al, a

mov a, Ahi

rlc a

mov Ahi, a

mov a,cycle

cjne a,#0,normalis\_next

inc razr\_num

ljmp normalis

normalis\_next:

dec cycle

ljmp normalis

Norm\_d: clr c

mov a, Bl

subb a, Al

mov Bl, a

mov a, Bhi

subb a, Ahi

mov Bhi, a

push psw

clr c

mov a, Cl

rlc a

mov Cl, a

mov a, Chi

rlc a

mov Chi, a

pop psw

jc next3

mov a, Cl

add a, #1

mov Cl, a

ljmp next3\_1

next3: clr c

mov a, Al

add a, Bl

mov Bl,a

mov a, Ahi

addc a, Bhi

mov Bhi,a

next3\_1: clr c

mov a, Ahi

rrc a

mov Ahi, a

mov a, Al

rrc a

mov Al, a

mov a,razr\_num

cjne a,#0, next4

divide\_ret: ret

next4:

dec razr\_num

ljmp Norm\_d

indicateA: mov dptr, #data

mov signs, Ahi

mov a, Ahi

anl a, #128

cjne a, #128, ia1

mov a, #45

movx @dptr, a

inc dptr

mov a, Ahi

anl a, #127

mov Ahi, a

ia1: push Ahi

push Al

push Bhi

push Bl

mov Bhi, Ahi

mov Bl, Al

push Bhi

push Bl

mov Ahi, #27h

mov Al, #10h

lcall divide

mov a, Cl

add a, #30h

movx @dptr, a

inc dptr

mov Bhi,#0

mov Bl,Cl

mov Ahi, #27h

mov Al, #10h

lcall Mult

mov Ahi, Chi

mov Al,Cl

pop Bl

pop Bhi

lcall Subb\_calc

mov Bhi, Chi

mov Bl, Cl

push Bhi

push Bl

; Tisiachi

mov Ahi, #03h

mov Al, #0E8h

lcall divide

mov a, Cl

add a, #30h

movx @dptr, a

inc dptr

mov Bhi,#0

mov Bl,Cl

mov Ahi, #03h

mov Al, #0E8h

lcall Mult

mov Ahi, Chi

mov Al,Cl

pop Bl

pop Bhi

lcall Subb\_calc

mov Bhi, Chi

mov Bl, Cl

push Bhi

push Bl

; Sotni

mov Ahi, #00h

mov Al, #64h

lcall divide

mov a, Cl

add a, #30h

movx @dptr, a

inc dptr

mov Bhi,#0

mov Bl,Cl

mov Ahi, #00h

mov Al, #64h

lcall Mult

mov Ahi, Chi

mov Al,Cl

pop Bl

pop Bhi

lcall Subb\_calc

mov Bhi, Chi

mov Bl, Cl

push Bhi

push Bl

; Desiatki

mov Ahi, #00h

mov Al, #10

lcall divide

mov a, Cl

add a, #30h

movx @dptr, a

inc dptr

mov Bhi,#0

mov Bl,Cl

mov Ahi, #00h

mov Al, #10

lcall Mult

mov Ahi, Chi

mov Al,Cl

pop Bl

pop Bhi

lcall Subb\_calc

mov Bhi, Chi

mov Bl, Cl

; Edinici

mov a, Bl

add a, #30h

movx @dptr, a

inc dptr

mov a, #32

movx @dptr, a

inc dptr

movx @dptr, a

inc dptr

movx @dptr, a

pop Bl

pop Bhi

pop Al

pop Ahi

mov Ahi, signs

lcall indic

ret

Subb\_calc:

clr c

mov a, Bl

subb a, Al

mov Cl, a

mov a, Bhi

subb a, Ahi

mov Chi, a

ret

sub\_abs: clr c

mov a, Ahi

cjne a, Bhi, NE1

mov a, Al

cjne a, Bl, NE1

mov Chi, #0

mov Cl, #0

ret

NE1: jc ALessB

AMoreB: clr c

mov a, Al

subb a, Bl

mov Cl, a

mov a, Ahi

subb a, Bhi

mov Chi, a

setb c

ret

ALessB: clr c

mov a, Bl

subb a, Al

mov Cl, a

mov a, Bhi

subb a, Ahi

mov Chi, a

clr c

ret

add\_sign: mov signs, #0

mov a, Ahi

anl a, #128

cjne a, #128, add\_sign1

mov signs, #1

add\_sign1:mov a, Ahi

anl a, #127

mov Ahi, a

mov a, Bhi

anl a, #128

cjne a, #128, add\_sign2

mov a, signs

orl a, #2

mov signs, a

add\_sign2:mov a, Bhi

anl a, #127

mov Bhi, a

mov a, signs

cjne a, #0, add\_sign3

lcall Addd

mov a, OVF

cjne a, #1, add\_s21

ret

add\_s21: mov a, Chi

anl a, #128

cjne a, #128, add\_s22

mov OVF, #1

ret

add\_s22: mov Ahi, Chi

mov Al, Cl

ret

add\_sign3:cjne a, #3, add\_sign4

lcall Addd

mov a, OVF

cjne a, #1, add\_s31

ret

add\_s31: mov a, Chi

anl a, #128

cjne a, #128, add\_s32

mov OVF, #1

ret

add\_s32: mov a, Chi

orl a, #128

mov Chi, a

ret

add\_sign4:lcall sub\_abs

jc AMB

ALB: mov a, signs

anl a, #2

cjne a, #0, SetSign

ret

AMB: mov a, signs

anl a, #1

cjne a, #0, SetSign

ret

SetSign: mov a, Chi

orl a, #128

mov Chi, a

ret

include C:\SHELL51\ASMS\4081\_3\Dorof\klavinc.asm

include C:\SHELL51\ASMS\4081\_3\Dorof\lcdinc.asm

*Листинг 3. Код файла lcdinc.asm*

indic\_w0: equ 20h

indic\_w1: equ 21h

cnt: equ 22h

indic:

indic\_init: mov indic\_w1,#0

mov indic\_w0,#38h

lcall indic\_wr

mov indic\_w0,#0Ch

lcall indic\_wr

mov indic\_w0,#80h

lcall indic\_wr

mov indic\_w1,#1

mov dptr,#data

mov cnt,#0

indic\_data\_wr\_1:

movx a,@dptr

mov indic\_w0,a

lcall indic\_wr

inc dptr

inc cnt

mov a,cnt

cjne a,#20,indic\_data\_wr\_1

mov indic\_w1,#0

mov indic\_w0,#0C0h

lcall indic\_wr

mov indic\_w1,#1

indic\_data\_wr\_2

movx a,@dptr

mov indic\_w0,a

lcall indic\_wr

inc dptr

inc cnt

mov a,cnt

cjne a,#40,indic\_data\_wr\_2

indic\_exit: ret

indic\_wr: mov 0E8h,indic\_w0

setb p1.7

clr p1.4

mov a,indic\_w1

mov c,acc.0

mov p1.6,c

lcall indic\_delay

clr p1.7

lcall indic\_delay

setb p1.7

ret

indic\_delay:

nop

nop

nop

nop

nop

nop

nop

nop

ret

data: db ' '

*Листинг 4. Код файла klavinc.asm*

map\_start: equ 30h

p5: equ F8h

klav\_num: equ 40h

klav: mov a, #EFh

mov r0, #map\_start

cycle2: mov p5, a

orl p5, #0Fh

mov @r0, p5

rl a

inc r0

cjne r0,#(map\_start+4),cycle2

mov r0, #map\_start

convert:

mov r2,#0

mov a,@r0

conv\_cyc:

rrc a

inc r2

jnc end\_conv

cjne r2,#4,conv\_cyc

inc r0

cjne r0,#(map\_start+4),convert

mov klav\_num,#20

jmp end

end\_conv:

mov a, r0

subb a,#30h

rl a

rl a

add a,r2

mov klav\_num,a

end: ret

1. Выводы

При проектировании этого проекта были полностью закреплены навыки программирования МК SAB 80C515 с подключенными модулями, накопленные за время выполнения лабораторных работ, и применены на практике теоретические знания об организации МК. Т.к. это задание является завершающим - то в нем по возможности были использованы разнообразные модули, такие как - клавиатура, ЖКИ дисплей.

Результатом данного индивидуального задания стал работоспособный простейший калькулятор.