НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ УКРАИНЫ

«КИЕВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ»

ФАКУЛЬТЕТ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Модульная контрольная работа № 1

«Архитектура компьютеров»

Выполнил: студент 3 курса ФИВТ
Группы ИВ-73
Ашаев Юрий Николаевич

Киев-2010

Задание

Тема: Выполнение операций сложения и вычитания с плавающей запятой в МК 51.

Цель работы: Изучение структуры памяти МК51, системы команд, форматов подачи данных и способов адресации операндов, получение навыков разработки программ выполнения простых арифметических операций над числами с плавающей запятой для МК 51.

Теоретические сведения

Суммирование и вычитание чисел в формате с плавающей запятой

Сумма двух чисел $X = 2^{P_x} M_x$ и $Y = 2^{P_Y} M_Y$, представленных в формате с плавающей запятой можно записать в виде:

$$2^{P_X} M_X + 2^{P_Y} M_Y = 2^{P_Z} M_Z$$

Для сложения чисел с плавающей запятой необходимо привести их к общему порядку P_z, в качестве которого лучше выбрать больший порядок из двух $P_z = \max(P_X, P_Y)$.

Во время этого уменьшения за счёт сдвига вправо мантиссы числа с меньшим порядком. В противном случае возникнет переполнение разрядной сетки мантиссы числа, которое преобразуется. После этого сумму чисел можно подать в виде:

$$2^{P_X} M_X + 2^{P_Y} M_Y^{/} = 2^{P_Z} (M_X + M_Y^{/})$$

Выполнение операции сложения или вычитания чисел с плавающей запятой в общем виде можно состоит из следующих этапов:

1) Выравнивание порядков.
2) Сумма мантисс.
3) Определение порядка результата.
4) Нормализация результата.
5) Округление результата.
6) Конечная нормализация результата.

Формат числа с плавающей запятой.

Для реализации арифметических операций с плавающей запятой в МК51 числа подаются в виде 32 разрядного двоичного кода, где один байт отвечает за порядок числа и 3 за мантиссу числа.

Симметричный порядок подаётся в положительном коде и изменяется (-128) –(127), где старший разряд знаковый. Смещённый порядок использует положительное число без знака от 0 до 255(нулевой порядок – сдвиг +126)

Выполнение:

Номер зачётки: $(7301)_{10} = (1\ 1100\ 1000\ 0101)_2$

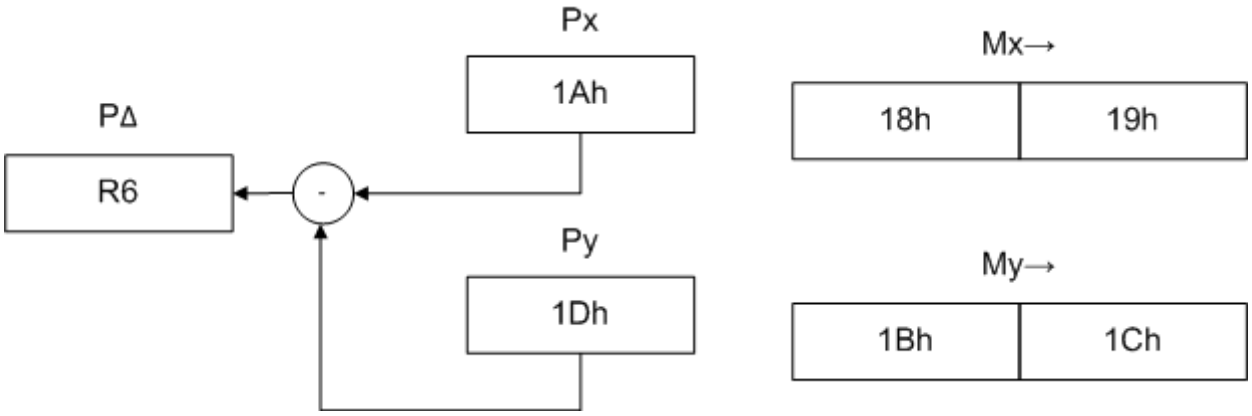
h1=1 – Операция **вычитание**, длина мантиссы – **2 байта**.

h4=0 – Формат подачи мантиссы – **ПК**.

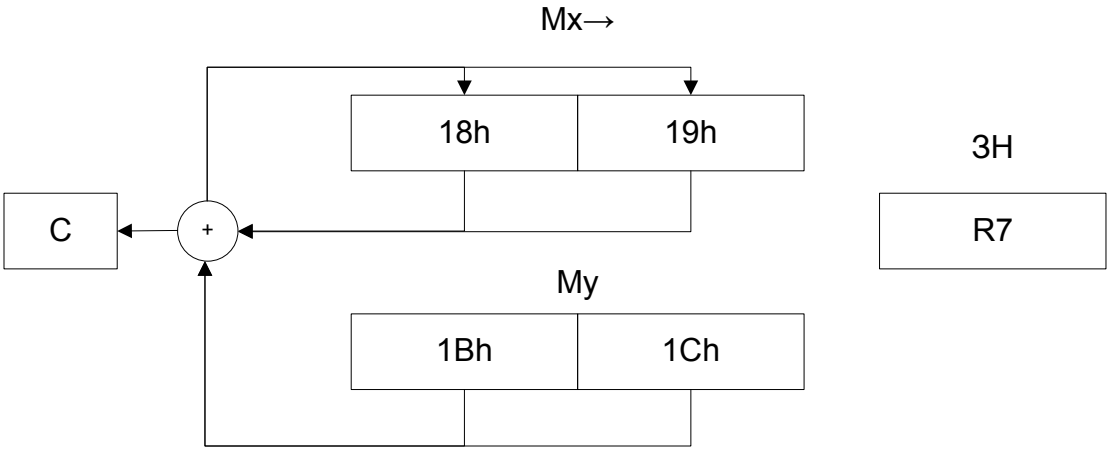
h2=0 – Формат подачи порядка **– Симметричный**

h5=0 – Первый операнд, результат – **РПД**; Второй операнд – **ЗПД**

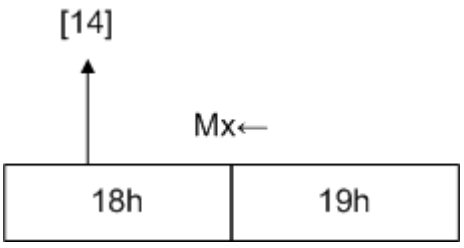
Операционная схема операции выравнивания порядков



Операционная схема операции сложения мантисс

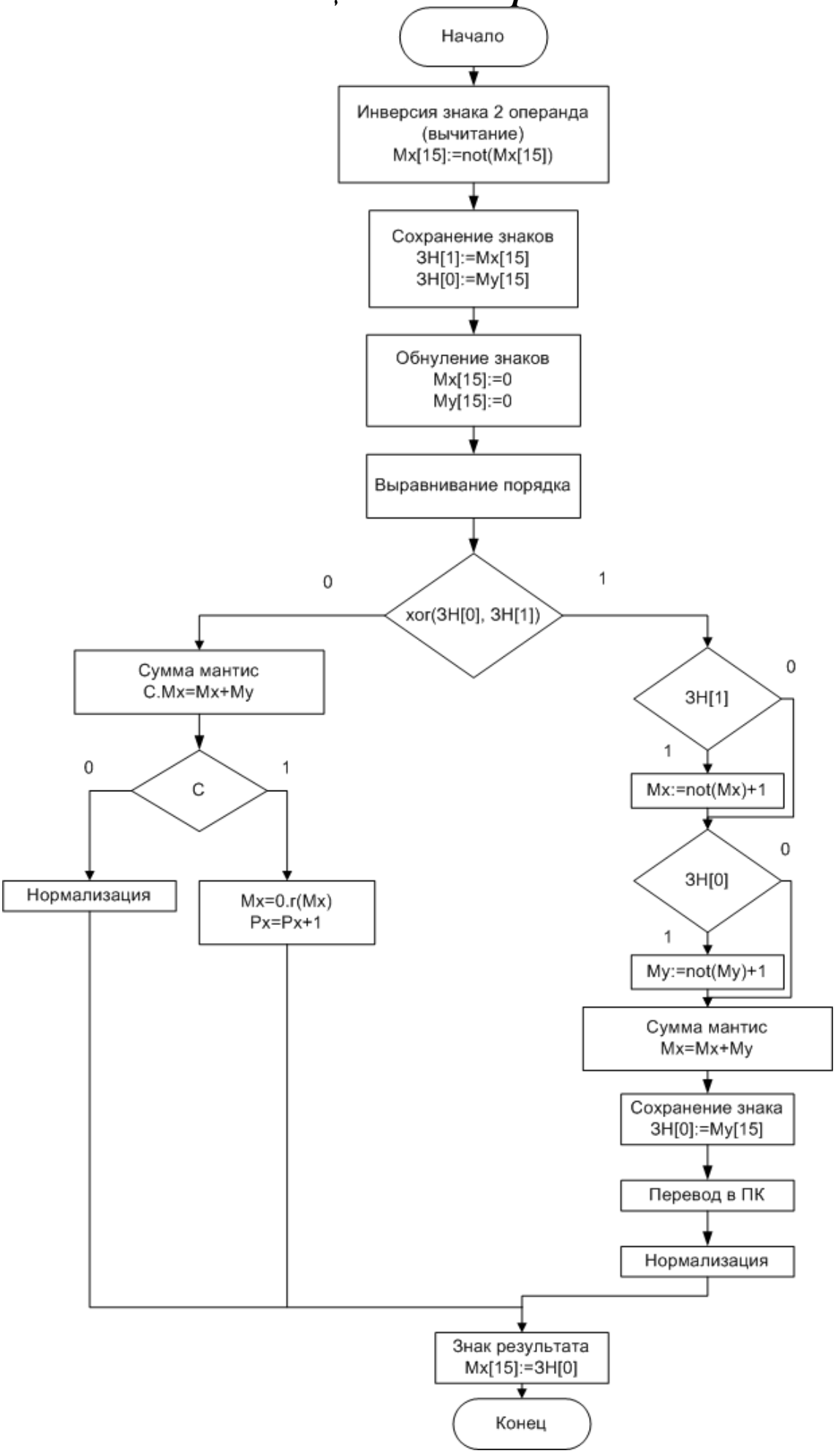


Операционная схема операции нормализации

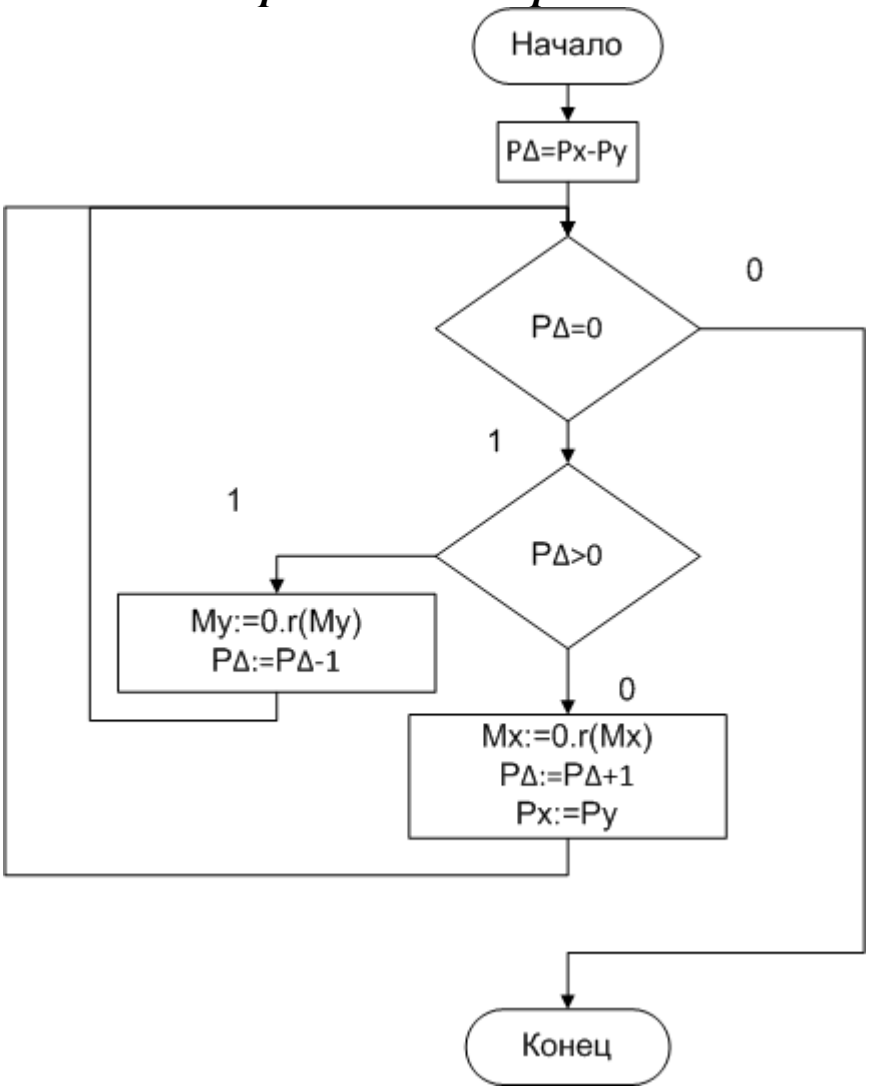


Описание. Мантиссы чисел хранятся по адресам 18h.19h для первого операнда и 1Bh.1Ch для второго. Порядки соответственно по адресам 1Ah и 1Dh. Рабочий регистр R7 хранит значения знаков операндов, а R2разность порядков. При нормализации чисел анализируется 14 разряд результата, который помещается по адресу первого операнда. Результат хранится в ПК.

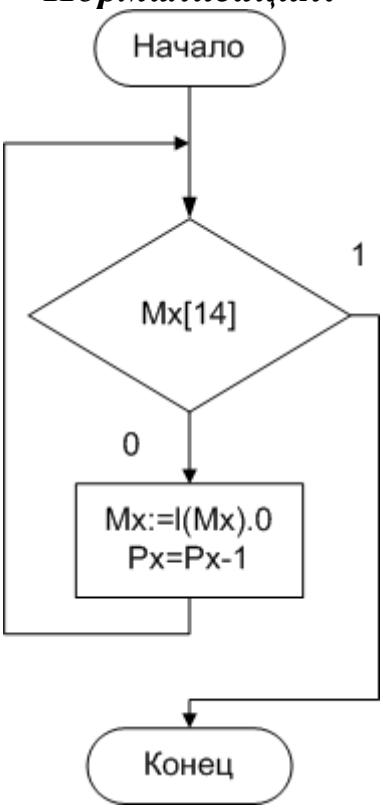
Обобщенный алгоритм



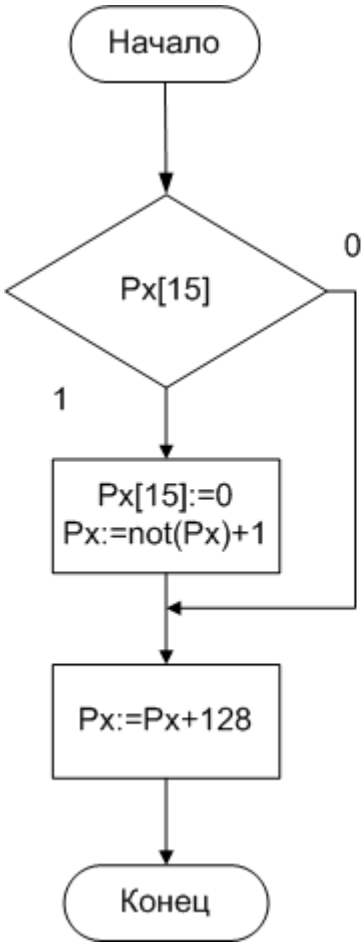
Выравнивание порядка:

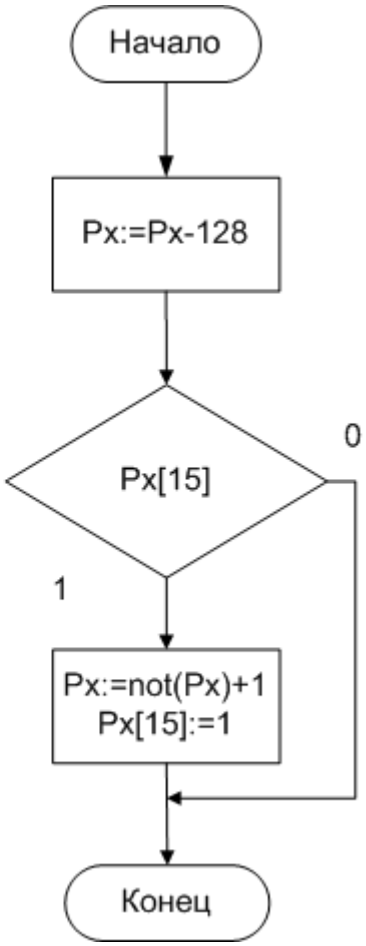


Нормализация:



Преобразование симметричного порядка (прямой код) в смещенный и обратно:





Листинг

```
; первый операнд
mov @18h, #03h
mov @19h, #47h
mov @1Ah, #3h

; второй операнд
mov @1Bh, #04h
mov @1Ch, #45h
mov @1Dh, #5h

; перевод порядков в
; смещенный код
mov r0, @1Ah
acall SIMSM

mov r0, @1Dh
acall SIMSM

; инверсия знака 2 операнда
; сохранение знаков
mov a, @18h
rlc a
mov a, #0
rlc a
mov r7, a
mov a, @1Bh
rlc a
mov a, r7
cpl c
rlc a
mov r7, a

; обнуление знаков
mov a, @18h
anl a, #7Fh
mov @18h, a

mov a, @1Bh
anl a, #7Fh
mov @1Bh, a

; разность порядков
mov a, @1Ah
clr c
subb a, @1Dh
mov r6, a

; сдвиг меншей мантиссы
mov r2, a
mov r6, a
shMant:
  mov a, r6
  anl a, #FFh
  jz endShM

  mov a, r2
  jb acc.7, shMx

    mov r0, #1Bh
    clr c
```

```
    acall SHR2
    dec r6
    ajmp shMant

  shMx:
    mov r0, #18h
    clr c
    acall SHR2
    inc r6
    mov a, @1Dh
    mov @1Ah, a
    ajmp shMant

endShM:

; проверяем равенство знаков
mov a, r7
anl a, #1h
mov r2, a

mov a, r7
clr c
rrc a
xrl a, r2

jnz nequ
  ; сумма мантисс
  mov r0, #19h
  mov r1, #1Ch
  acall SUM2

  ; было ли переполнение
  jc overf
    mov r0, #18h
    acall NORM2
    ajmp sing
  overf:
    mov r0, #18h
    acall SHR2
    mov a, @1Ah
    inc a
    mov @1Ah, a
    ajmp sing
; если знаки разные
nequ:
  mov a, r7
  mov r0, #19h
  jb acc.1,mxNeg
    ; если My отрицательное
    mov r0, #1Ch
  mxNeg:
  acall DKPK2

  ; сумма мантисс
  mov r0, #19h
  mov r1, #1Ch
  acall SUM2

  ; сохранение знака
  mov a, @18h
  rlc a
  mov a, r7
  rlc a
  mov r7, a

  ; перевод в ПК
  mov a, #18h
  jb acc.7,resPos
    ; если результат отриц.
    ; перевод в ПК
    mov r0, #19h
    acall DKPK2
    ajmp sing
  resPos:
  mov r0, #18h
  acall NORM2

; знак результата
sing:
  mov a, r7
  jnb acc.1, nSign
    mov a, @18h
    mov r2, #80h
    orl a, r2
    mov @18h, a
  nSign:

; перевод смещенного порядка
; результата в симметричный
mov r0, @1Ah
acall SMSIM

ajmp exit

; Сдвиг 2 байтов влево с адресами
; r0+1 r0
; вдвигается C
SHR2:
  mov r1, #2h
  shr2loop:
    mov a, @r0
    rrc a
    mov @r0, a
    inc r0
  djnz r1, shr2loop
ret

; Сумма 2 байтов
; r0+1 r0
; r1+1 r1
; резульат r0+1 r0
SUM2:
  mov r2, #2h
  clr c
  sum2loop:
    mov a, @r0
    addc a, @r1
    mov @r0, a
    dec r0
    dec r1
  djnz r2, sum2loop
ret

; Нормализация мантиссы 2 байта
; r0+1 r0
NORM2:
  mov a, r0
  mov r1, a
  inc r1
  mov r2, 16
  norm2l:
    mov a, @r0
    jb acc.6, exNorm
    clr c
    mov a, @r0
    rlc a
```

```
    mov @r0, a
    mov a, @r1
    rlc a
    mov @r1, a
  djnz r1, norm2l
exNorm: ret

; Перевод в ДК <-> ПК
; r0+1 r0
; r1 - вспомогательный
DKPK2:
  mov r1, #2h
  clr c
  cpl c
  DKPK2l:
    mov a, @r0
    cpl a
    addc a, #0h
    mov @r0, a
    dec r0
  djnz r1, DKPK2l
ret

; Перевод симметричного порядка в
смещенный
; r0 - адрес
SIMSM:
  mov a, r0
  jb acc.7, simsml
  anl a, #7Fh
  cpl a
  inc a
  simsml:
  add a, #80h
ret

; Перевод смещенного порядка в
симметричный
; r0 - адрес
SMSIM:
  mov a, r0
  clr c
  subb a, #80h
  jb acc.7, smsiml
  cpl a
  inc a
  orl a, #80h
  smsiml:
ret

exit: end
```

Тема: Выполнение сложных арифметических операций с плавающей запятой в МК 51.

Цель работы: Изучение структуры памяти МК51, системы команд, форматов подачи данных и способов адресации операндов, получение навыков разработки программ выполнения сложных арифметических операций над числами с плавающей запятой для МК 51.

Теоретические сведения

Добывание квадратного корня из числа с плавающей запятой

Добывание квадратного корня из числа $X = 2^{P_x} M_x$, что задано в формате с плавающей запятой, можно подать в виде:

$$Y = \sqrt{X} = \sqrt{2^{P_x} M_x} = 2^{\frac{P_x}{2}} \sqrt{M_x}$$

Таким образом, для получения квадратного корня из числа с плавающей запятой нужно порядок числа поделить на два, а с мантиссы получить квадратный корень по правилам для чисел с фиксированной точкой.

Деление порядка на два производится путем сдвига его на разряд вправо, если порядок четный. Если порядок нечетный, то к нему нужно добавить единицу и сдвинуть мантиссу на один разряд вправо, после чего прядок сдвигают на один разряд вправо – деление на два. То есть, если порядок $P_x = 2k - 1$, то

$$X = 2^{2k-1} M_x = 2^{2k} (M_x \times 2^{-1}), \quad Y = \sqrt{X} = 2^k \sqrt{M_x \times 2^{-1}}$$

Мантиссы числе с плавающей запятой всегда нормализованные и , когда в первом цикле из мантиссы происходит отнимание числа 0,01, то первый остаток будет всегда положительным, таким образом первая цифра результата будет всегда равняться единице. В соответствии с этим, при выполнении операции получения квадратного корня в устройстве с плавающей точкой не может быть нарушения нормализации.

Этапы получения квадратного корня из числа с плавающей запятой следующие: получение порядка результата; получение мантиссы результата, которая всегда нормализована, поэтому этап нормализации не выполняется.

Выполнение:

Номер зачётки: $(7301)_{10} = (1\ 1100\ 1000\ 0101)_2$

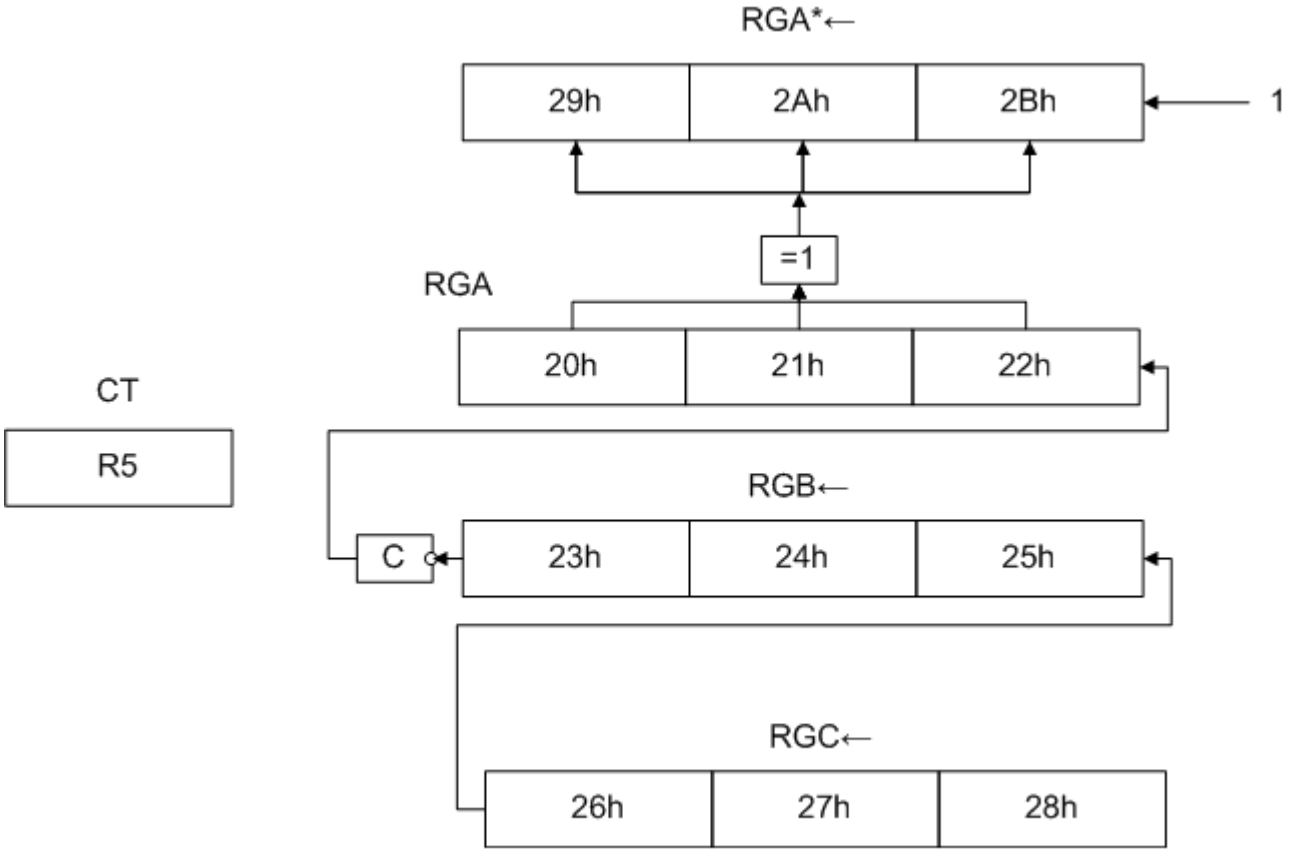
h0h1=10 – Операция **извлечение корня**, длина мантиссы – **3 байта**.

h4=0 – Формат подачи мантиссы – **ДК**.

h2=1 - Формат подачи порядка **– Симметричный**

h5=0- Первый операнд, результат- **ЗПД**; Второй операнд - **РПД**

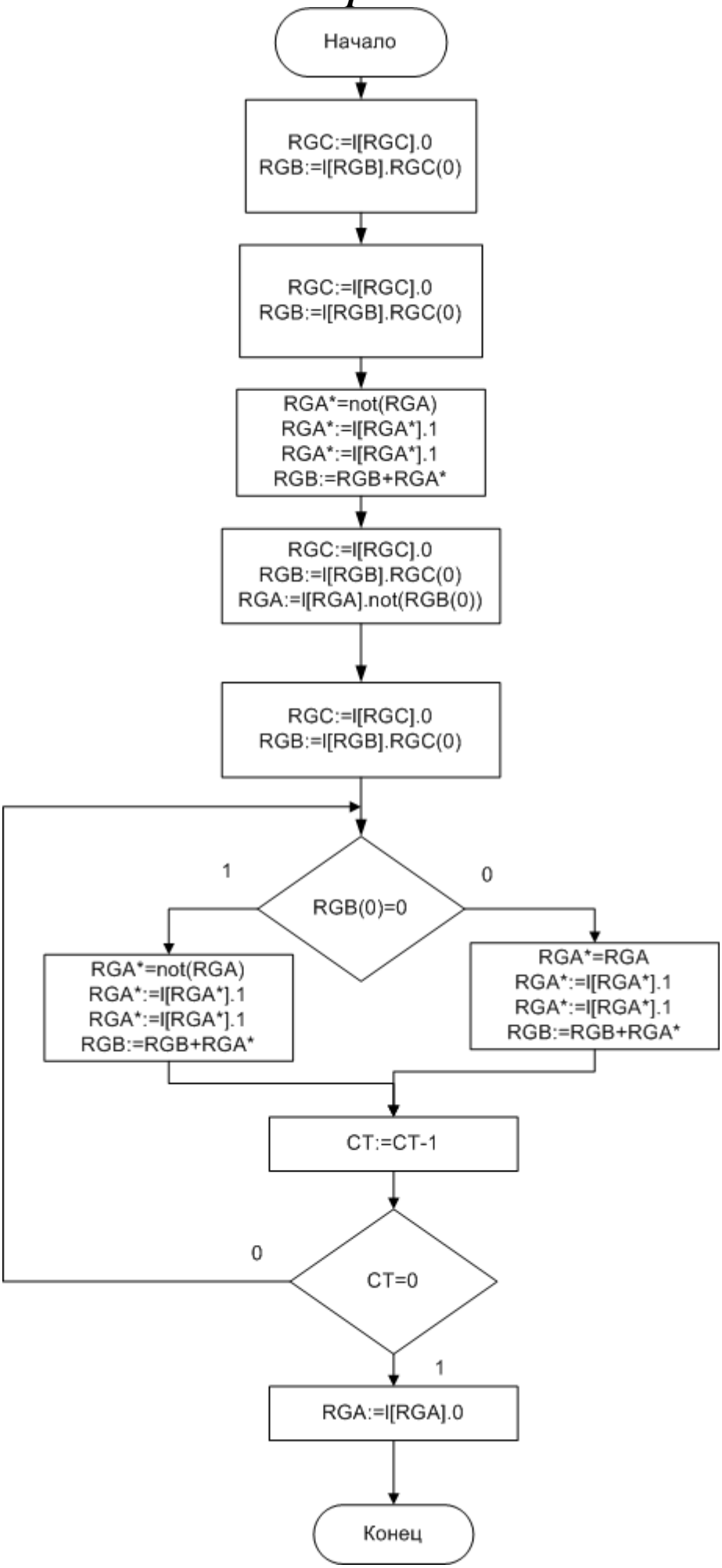
Операционная схема получения цифр результата и операции получения вспомогательного значения



Операционная схема получения промежуточного остатка

Описание. Регистры находятся по соответствующим адресам памяти. Регистр RGC содержит беззнаковое делимое. Регистр RGA используется для накопления цифр результата. RGA* необходим для получения инвертированного или неинвертированного значения регистра RGA, сдвинутого на 2 разряда влево с заполнением вдвигаемых разрядов единицами. RGB хранит промежуточное значение остатка.

Алгоритм


Начало
RGC:=I[RGC].0
RGB:=I[RGB].RGC(0)
RGC:=I[RGC].0
RGB:=I[RGB].RGC(0)
RGA*=not(RGA)
RGA*:=I[RGA*].1
RGA*:=I[RGA*].1
RGB:=RGB+RGA*
RGC:=I[RGC].0
RGB:=I[RGB].RGC(0)
RGA:=I[RGA].not(RGB(0))
RGC:=I[RGC].0
RGB:=I[RGB].RGC(0)
1
RGB(0)=0
0
RGA*=not(RGA)
RGA*:=I[RGA*].1
RGA*:=I[RGA*].1
RGB:=RGB+RGA*
RGA*=RGA
RGA*:=I[RGA*].1
RGA*:=I[RGA*].1
RGB:=RGB+RGA*
CT:=CT-1
0
CT=0
1
RGA:=I[RGA].0
Конец

Листинг

```
mov @81h, #30h

;исходное число
mov @26h, #00010001B
mov @27h, #00110110B
mov @28h, #00011010B
;порядок
mov @2Dh, #4h

;формирование порядка
;результата
mov a, @2Dh
anl a, #1h

jnz mainl5

clr c
mov r0, #28h
acall SHL3

mainl5:

mov a, @2Dh
add a, #1h
rrc a
mov @2Dh, a
;сдвиг RGB RGC влево
;------------------------
clr c
mov r0, #28h
acall SHL3

mov r0, #25h
acall SHL3
;сдвиг RGB RGC влево
;------------------------
clr c
mov r0, #28h
acall SHL3

mov r0, #25h
acall SHL3
;формирование RGA*
;------------------------
mov r0, #22h
mov r1, #2Bh
acall COPY3

mov r0, #2Bh
acall INV3

mov r2, #2h
mainl1:
clr c
cpl c
mov r0, #2Bh
acall SHL3
djnz r2, mainl1

;получение суммы RGB:=RGB+RGA*
mov r0, #25h
mov r1, #2Bh
acall SUM3
;------------------------

mov r5, #16h
```

```
mainl2:

;сдвиг RGB RGC влево
;------------------------
clr c
mov r0, #28h
acall SHL3

mov r0, #25h
acall SHL3

;вдвигание цифры корня в RGA
cpl c

mov r0, #22h
acall SHL3

;сдвиг RGB RGC влево
;------------------------
clr c
mov r0, #28h
acall SHL3

mov r0, #25h
acall SHL3
;формирование RGA*
;------------------------
mov r0, #22h
mov r1, #2Bh
acall COPY3

mov a, @23h
rlc a

jc mainl3

;инверсия RGA*
mov r0, #2Bh
acall INV3

mainl3:

mov r2, #2h
mainl4:
clr c
cpl c
mov r0, #2Bh
acall SHL3
djnz r2, mainl4

mov r0, #25h
mov r1, #2Bh
acall SUM3

;CT:=CT-1
djnz r5, mainl2

mov r0, #22h
acall SHL3

ljmp exit

; Сдвиг 3 байтов влево с адресами
; r0+2 r0+1 r0
; r1 - вспомогательный
```

```
; вдвигается C
SHL3:
  mov r1, #3h
  shl3loop:
    mov a, @r0
    rlc a
    mov @r0, a
    dec r0
  djnz r1, shl3loop
ret

; Инверсия 3 байтов с адресами
; r0+2 r0+1 r0
; r1 - вспомогательный
INV3:
  mov r1, #3h
  inv3loop:
    mov a, @r0
    cpl a
    mov @r0, a
    dec r0
  djnz r1, inv3loop
ret

; Копия 3 байтов с адресами
; r0+2 r0+1 r0 в
; r1+2 r1+1 r1
; r2 - вспомогательный
COPY3:
  mov r2, #3h
  copy3l:
    mov a, @r0
    mov @r1, a
    dec r0
    dec r1
  djnz r2, copy3l
ret

; Сумма 3 байтов с адресами
; r0+2 r0+1 r0 и
; r1+2 r1+1 r1
; r2 - вспомогательный
; результат r0+2 r0+1 r0
SUM3:
  mov r2, #3h
  clr c
  sum3l:
    mov a, @r0
    addc a, @r1
    mov @r0, a
    dec r0
    dec r1
  djnz r2, SUM3l
ret

exit: end
```