1. Архитектура

1.1. Регистровая память

Регистры SFR используются с двумя принципами доступа – прямой доступ и адресный в RAM.

- a(Acc) – основной регистр-аккумулятор, применяемый во всех арифметических и логических операциях с прямым доступом (a) – mov a,r0. В команде mov Acc,r0 подразумевается адресный доступ к теневому регистру в Ram
- B – рабочий регистр, неявно доступен в командах умножения mul ab и деления div ab или по адресу в SFR – mov b,r0.
- Регистр состояния PSW=C.AC.F0.RS1.RS0.OV.-.P содержит признаки результата арифметических операций – C(перенос, заем), AC – полуперенос, OV(переполнение), P(бит четности), F0(бит пользователя), RS1-RS0 – номер активного регистрового банка – неявно используется в арифметических операциях и доступен по адресу в команде mov a, PSW
- PC – 16-разрядный программный счетчик или регистр адреса сполняемой команды. При включении питания автоматически сбрасывается. Таким образом, в MCS51 начальный запуск программы с адреса 0000. PC адресного доступа не имеет, но может контролироваться косвенно и модифицируется неявно.
- DPTR – 16-разрядный адресный регистр (Data Pointer) обращения к внешней памяти программ Code и данных Xdata. Доступен по адресам образующих его 8-битовых регистров DPTR=DPH.DPL в SFR.

1.2. Иерархическая оперативная память данных Ram

Иерархическая оперативная память данных Ramc прямой адресацией включает память Data и SFR. Память Data включает Stack, регистровые банки памяти Ri , битовую память Bit Все представленные типы памяти различаются способом доступа к данным (прямой или косвенный) и типом данных (бит, байт), но могут относиться к одним и тем же ячейкам Ram.

1.2.1. Сегмент Data

В ассемблере порядок размещения находится под контролем программиста и может быть изменен, кроме порядка, закрепленного аппаратно – адресация битов, адресация регистров в четырех банках, начальное размещение стека.

1.2.2. РОН

Регистры общего назначения Ri={ R0,R1,..R7 } (регистровая адресация) – активный регистровый банк. В C51 нулевой банк резервирован для рабочих регистров. Доступны 4 банка, совмещенные с начальными ячейками памяти

Data. Активный банк выбирается в регистре PSW. Адрес соответствующей ячейки Data определяется смещением относительно банка (RS1.RS2).Ri (например, в 3-ем банке регистр R2 имеет адрес 0x1A) Доступ к этим ячейкам регистровый.

```
mov  a,R0   ; Ram[R0]-> Acc
mov R1,a    ; Acc -> Ram[R1]
```

Прямой доступ в Data может быть полезен, так как отсутствует двух адресная регистровая команда для передачи данных между регистрами – вместо этого, например, может быть

```
mov  00, r1
mov  r3, 01
mov  07,1a
```

Косвенная регистровая адресация

```
mov R0, #Dm  ; регистр косвенной адресации
mov a, @R0   ; косвенная адресация
inc R0
mov Dm, @R0  ; Ram[R0] -> Ram[Dm]
```

1.2.3. Регитры SFR

Регистры SFR с прямой адресацией в Ram (80-FFh), 128 байт – управляющие и системные регистры. К SFR относятся указатель стека SP, таймеры TH0, TL0, TH1,TL1, регистры ACC, B, PSW, DPTR=DPH.DPL, регистры портов P0, P1, P2, P3.

1.2.4. Сегмент битов

Bit – 128 бит, прямой адрес бита 0x0-0x7f, память совмещена с ячейками 0x20-0x2f в Data, где i-ый бит находится в ячейке Data с адресом 0x20+i/8, номер бита i%8.

```
bseg at 0x10  ;сегмент битов с 0x10-го бита в поле бит Data
x0: dbit 4    ;поле из четырех бит в сегменте
```

В SFR биты индексируют $i \in [0;7]$ разряды бит-доступных регистров и не входят в сегмент битов с последовательной адресацией.

```
x4 bit ACC.5  ; битовая переменная, соответствующая 5-ому биту ACC
mov c, 0      ; Data(20h.0) -> C , 20h.0 -- нулевой бит ячейки Data
mov ACC.7, c  ; c -> Acc.7,
mov c, x0+2   ; x0 -- адрес первого бита поля бит
mov x4,c
```

1.2.5. Стек

Stack - в mcs51 используются ячейки памяти Data. К стеку возможно как явное обращение, так и неявное – в прерываниях и переходах к подпрограммам.

Явное обращение к стеку:

```
push ad     ; запись байта в стек
```

Например, push Acc обозначает Ram[Acc] → Data[+SP].

```
pop ad      ; чтение байта из стека
```

Например, pop Acc обозначает Data[SP–] → Ram[Acc].

При включении питания и сбросе в MCU устанавливается по умолчанию SP=07.

1.2.6. Сегмент программной памяти Code

Доступ к данным:

```
mov a,#d      ; Code[PC+] -> a    -- непосредственная адресация
movc a,@a+pc  ; Code[PC + a] -> a -- адресация относительно
              ; текущего PC, в ACC размещается индекс
mov dptr,#yy  ; сохранение адреса
movc a,@a+dptr ; Code(dptr + a) -> a, базовая адресация:
               ; база в DPTR, в ACC смещение
```

1.2.7. Сегмент Xdata

```
mov dptr,#mm    ; адрес
movx a, @dptr   ; Xdata(dptr) -> A
movx  @dptr,a
```

1.3. Ввод-вывод в A51

Быстрый параллельный ввод-вывод осуществляется прямым обращением портам MCU. Порты содержат регистр данных, входные и выходные буферные схемы, подключаемые к внешним контактам MCU. При вводе (char x=P1) данные считываются с контактов порта и сохраняются в памяти, обычно интерпретируются в приложениях в положительном кодировании двоичными кодами (H 1, L 0). При выводе (P2=0x55) данные из памяти записываются в порт и передаются на внешние контакты.

1.4. Знаковая арифметика

Признаки C,OV,P в PSW, в скобках .. обозначены режимы адресации второго операнда.

```
add  a, {Ri,@rj,#d,ad} ;  a + {..}    -> a
addc a, {Ri,@rj,#d,ad} ;  a + {..} + C -> a
subb a, {Ri,@rj,#d,ad} ;  a - {..} - C -> a
```

1.5. Беззнаковая арифметика

```
inc {a, ri, @rj, ad, dptr}  ; {..} + 1, признаки не меняются в PSW.
dec r0, {a, ri, @rj, ad}    ; {..} - 1
mul ab ; a*b -> b.a, признаки: V=(b0), 0 -> C,  P.
div ab ; a/b -> a, b=rest(a/b),  признаки: OV, P.
rrc a, ; RR(c.a) -> (a.C), признаки C, P.
rlc a, ; RL(a.C) -> (C.a), признаки C, P.
clr a, ; 0 -> a
```

1.6. Десятичная арифметика

`DA a` – десятичная коррекция результатов двоичного сложения или вычитания 2/10 чисел.
`swap a` – обмен тетрадами в Acc.
`xchd a, @rj` – обмен тетрадами.

1.7. Логика поразрядная 8 битовая

```
anl a,  {Ri,@rj,#d,ad} ; a & {..} -> a, признаки: P, 0 -> C,
anl ad, {#d, a}
orl a,  {Ri,@rj,#d,ad} ; a $\lor$ {..} -> a,  признаки P, 0 -> C,
orl ad, {#d, a}
xrl {Ri,@rj,#d,ad}     ; a # {..} -> a, признаки P, 0 -> C
xrl ad, {#d, a}
cpl a ; not  a
rr a ; циклический сдвиг Acc вправо (признак C не изменяется)
rl a ; циклический сдвиг Acc влево (признак C не изменяется)
```

1.8. Битовые операции

```
anl c,{bit, /bit}   ; /bit - инверсия бита
orl c,{bit , /bit}
mov c,bit
```

```
setb bit,
clr bit,
cpl  C
```

1.9. Управление программой и ветвления

```
ljmp a16      ; PC -> a16
ajmp a11      ; PC(10.0) -> a11[10.0]
sjmp rel      ; PC+2+/- rel[6.0]
jmp  @a+dptr  ; PC -> a+dptr
jz   rel      ; PC+2+/-rel[6.0],если (a=0)
jnz  rel      ; если (a<>0)
jnc  rel      ; если !C
jb   bit,rel  ; PC+3+rel, если bit=1
jnb  bit,rel  ; если bit=0
jbc  bit,rel  ; если bit=1,bit<-0
djnz {ri,ad},rel     ; {}-1, PC+1/2+/-rel[6.0],если {}<>0
cjne {ri,@rj},#d,rel ; rel,если {}<>#d
lcall a16       ; стек -> pc, PC -> a16
acall a11       ; PC(10-0) -> a11[10.0]
ret             ; PC -> стек
reti            ; PC -> стек, tf -> 0
nop             ; пропуск
```