1) Принстоновская архитектура – признаками архитектуры являются одна шина доступа в ПЗУ и ОЗУ (одно адресное пространство), регистры имеют заранее определенное назначение.

Прерывание – это аппаратная реализация отклика синхронной системы на асинхронное событие.

3 способа реагирования: 1) игнорировать запрос 2) быстро среагировать на запрос 3) быстро среагировать и обслужить запрос полностью.

Командный цикл – МК состоит из нескольких тактов синхронизации, которые необходимы для выполнения команд. Некоторые команды могут требовать нескольких командных циклов, свойственно CISC.

Архитектура – общая конфигурация основных компонентов микропроцессора, их главные возможности и хар-ки, и взаимосвязь

Интерфейс – совокупность шин, сигналов, электр. схем и алгоритмов предназначенных для управления передачей информации между устройствами.

Микропроцессор – программно-управляемое устройство, осуществляющее процесс обработки цифровой информации и управления им, построенное на одной или нескольких интегральных схем.

2) Цикл, найти min элемент массива.

(8051)

min:

```
anl PSW,#(255-16?)
mov R0,#30h
mov R7,#30h
mov R6,#0
L-loop:
movA,@R0
clrc
subb A,R6
jnc L-next
mov 06h,@R0
L-next:
inc R0
djnz R7,L-loop
```

max:

```
anl PSW,#(255-16?)
mov R0,#30
mov R7,#30
mov R6,#0
L_loop:
mov A,@R0
clr
subb A,R6
jc L-next
mov 06h,@R0
L_next:
inc R0
djnz R7,L_loop
```

в R6 → max

(AVR)

min:

```
ldi R26,$60
Ldi R27,0
Ldi R18,$60+30
Ldi R16,X+
L-loop:
ld R17,X+
cp R17,R16
brbc (4) L-next
mov R16,R17
L-next
cpse R26,R18
rjmp L-loop
```

в R16 → min
4 бит SREG - знак знач.
brbc - переход на метку, если бит=0

max:

```
Ldi R26,$60
Ldi R27,0
Ldi R18,$60+30
Ldi R16,X+
L-loop:
ld R17,X+
cp R17,R16
brbs 4,L-next
mov R16,R16
L-next:
cpse R26,R18
rjmp L-loop
```

в R16 → max
4 бит SREG - знак рез-та
brbs - переход на метку если бит=1
cpse - пропуск след. команду если раз=0

1) массив длиной 30 байт располож. по адресу $30
2) R0 указ на текущ. элемент 3) R6 хранит max/min зн. масс.

Процедуры и функции с несколькими вх. перем.

(8051)

```
jmp start
org 0030h
start:
mov dpl,#50 // откуда 1 сегм (откуда)
mov dph,#$00 // куда копир.
push dpl
push dph
mov dpl,#10 // сколько копир
push dpl
call copy
copy: pop 00h // поместить
      pop 01h    знач. R0,R1,R2.
      pop 02h
metka:
mova,@r2 // откуда
mov @r1,a   копируем байт r2 → r1
inc r2
inc r1
djnz R0,metka ; копир. 10р.
reti
```

r0 = 10, r1 = 100, r2 = 50

(AVR)

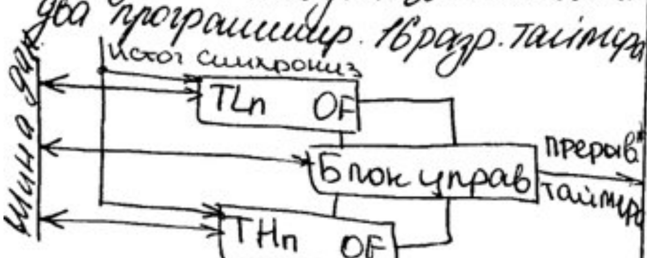
```
jmp main
main
   LDI R20,$60+30   ] постр
   out SPL,R20      ] стека на конец памяти
   LDI R26,$60 - откуда
   LDI R28,$80 - куда
   LDI R17,$0A - длина
push R26  }
push R28  } сохр. парам. в стек
push R17  }
call copy
copy:
   pop R17
   pop R28
   pop R26
   push R1
metka:
   LD R1,X+ - загр дан. из адр. $60 и $60+30
   ST Y+,R1 - загр в память по адр. $80 и $80+20
   dec R17
   BRBC 1,metka
   pop R1
ret
```

1) Ядро – базовое устройство внутр. вычисл. системы. Ядро определяет систему команд, шинный интерфейс, архитектуру памяти, т.е. главные отличия вычисл. систем друг от друга. Ядро МК м.б одинаковым, а изготовители разные.

2) Сравнение таймеров 16-разряд.

(8051)

В базовых модел. 8051 имеются два программир. 16разр. таймера



TLn и THn явл. 8раз. счетчиками – регистрами и представ. млад. и стар. разряды таймера; n-№ таймера

Ф-ии: – реализ. меток реал. вр – измер. числа событий за опредеn. промежуток вр. – измерение длительности импульса; – измер. частоты сигнала; – реализ. ШИМ

4 режима работы таймера:

Реж 0 – таймер конфигур. как 13раз счетчик

Реж 1 – -//- как 16разряд счетчик

Реж 2 – применяется для задания скорости последоп. порта.

Реж 3 – TLn и THn – работают как два независ. 8разряд счетчика. (только таймером 0)

Для управ таймерами/счет. предусмотр. 2 регистра: TMOD (089h) и TCON (088h)

TMOD:

7	6	5	4	3	2	1	0
Gate	C/T	M1	M0	Gate	C/T	M1	M0
тай/сч 1				тай/счет. 0			

Gate – управ. блокир. =0 – разр, если TCON.TRn-устав; =1 – -//-, если на вх внеш. пр. выс ур и TCON-уст

C/T – управ. тактир. =0 – работ внутр. генератора с делит 12; =1 – от сигнала на внеш. вх Tn.

M1.M0 – выбор реж. раб.
00 – реж 0 10 – реж 2
01 – реж 1 11 – реж 3

(AVR)

с точки зрения прог. упр:
– регистры управ (TCCR1A, TCCR1B)
– флаг переполн. и его маска прерыв (TIFR.TOV1, TIMSK.TOIE1)
– регистр вх. захвата (ICR1)
– флаг захвата и его маска прерыв
– регистр сравнения (OCR1A)
– флаг сравнения и его маска прерывания (TIFR.OCF1A, TIMSK.OCIE1A)

Возможности 1 шире 0. В частности с помощью встроен. регистра срав. и компаратора можно чтобы счетчик работал как делитель на любое. Таким образ. возможна реализация генератора с широко-импульс. модул (8, 9, 10бит) возм. измер. длительность внеш. процессов.

3) Цап с 11-ю ... AVR

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TCON: TFn – флаг переполн. счетчика
TRn – сигнал разр. раб. счетчика.

1) Гарвардская архитектура – признаками данной архитектуры является раздел шины доступа в ПЗУ и ОЗУ, регистры однородны по назначению (универсал. однотип.)

2) Чтение и запись данных и программ

(8051)

```
mov x <пр>, <источник>
```

пересылка в/из внеш. памяти

```
1 movx a, @Ri
2
3 movx a, @dptr
4
```

```
movc <пр>, <ист>
```

пересылка в(из) памяти прогр.

```
movc a, @a+dptr
movc a, @a+pc
```

(AVR)

LDS Rd, addr, загрузка байта из СОЗУ в регистр

STS addr, Rd, загрузка байта в СОЗУ из регистра

LD Rd, index, загрузка Rd байтом из памяти с адресом хранящимся в X, Y или Z

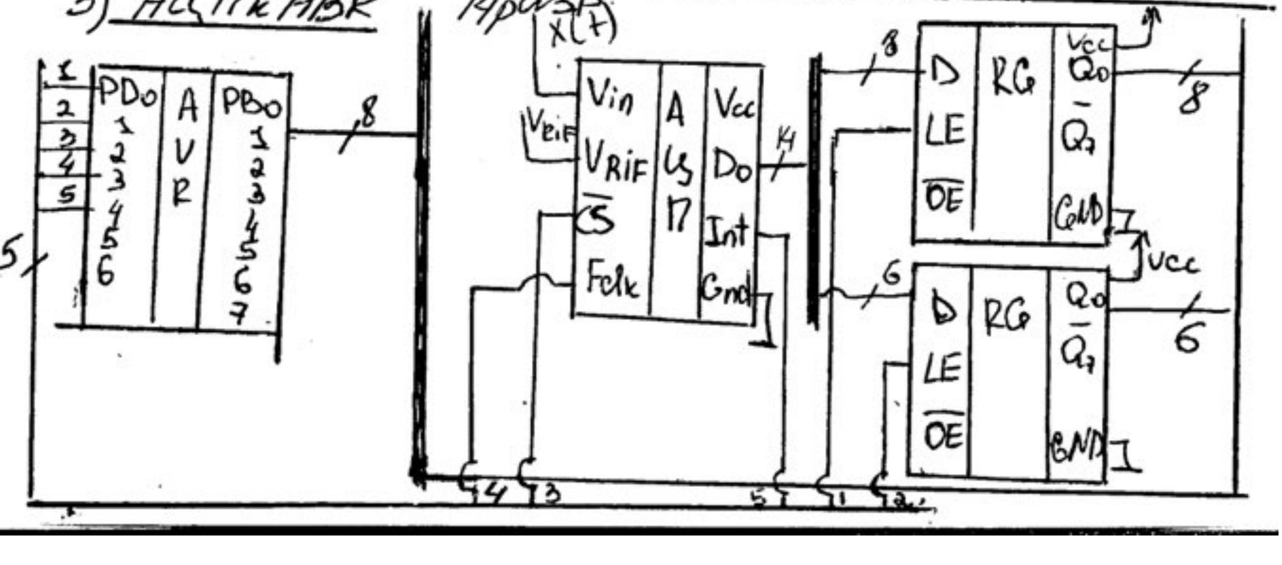
```
xseg
S1 ds 1
cseg
jmp start
org 0040h
start:
  mov dptr, #S1
  movx a, @dptr
call proc
N1 : db 7
proc:
    mov dptr, #N1
    clr a
  movc a, @a+dptr
```

Пример:

```
LDS r2, $FF00, загр. в r2 содер SRAM по адр.
STS $FF00, r2 ; загр байт из регист. в СОЗУ по адр.
LPM ; 1байт, адрес Z в рег. R0

clr r31, очист. стар. байт Z
LDI r30, $F0 ; уст. младш. байта Z
lpm ; загруз. конст. из ПП
```

3) АЦП к AVR



1) CISC — процессоры с полным набором инструкций. Состав и назначение их регистров существенно неоднородно. (кол-во рег. невелико) Команды создаются для удобства, а не эффективности исполнения. Это усложняет декодирование программы команд и как результат возрастает число тактов синхрон. необход. для выполн. команды, длины команд различны.

2) Процедура обработ. типов данных, хран. в пам. прогр.

8051

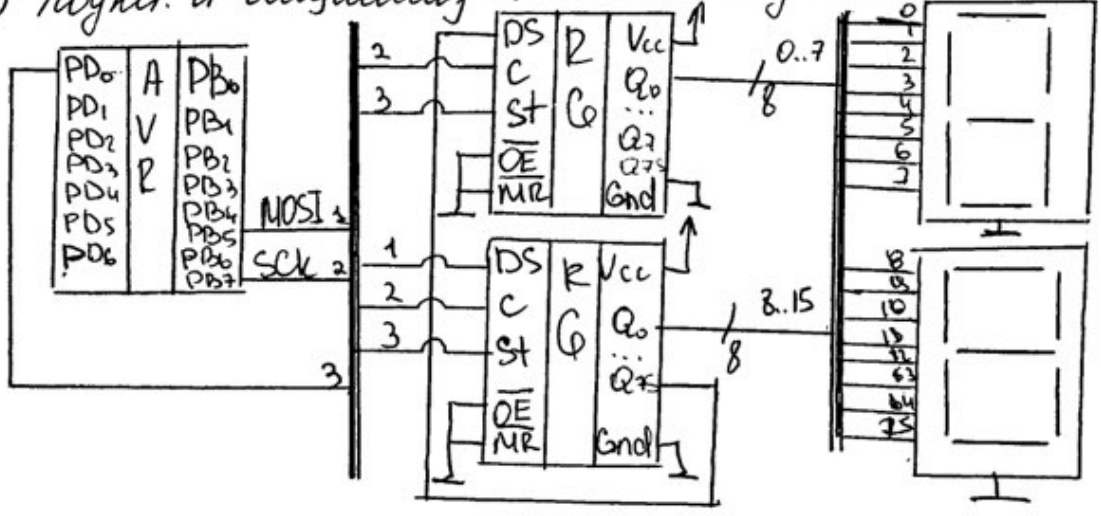
```
dseg at 30h
mass ds 5
cseg
 sjmp start
 org 0030h
 MdB 4,3,2,1,0
start
    mov R0,#mass
    mov dptr, #M
    mov R1, #5
    clr a
m1: movc a, @a+dptr
    mov @R0,a
    inc R0
    inc dptr
    djnc R1,m1
```

память програм-4кб-ПЗУ
память данных-128байт-ОЗУ

AVR

```
dseg
 NdB 5
 var: Byte 5
cseg
    ldi R26, Low(var)
    ldi R27, High(var)
    ldi R31, High(data.2)
    ldi R30, Low(data.2)
    ldi R17, N
ПЗУ-2кб ОЗУ-128байт
loop: LPM
  ST Xt, R0
   inc z
  dec R17
brbc 1, loop
rjmp exit
data: db 1,2,3,4,5
exit: end.
```

3) Подкл. и инициализ. семисег. инд. к АВР



1) Сторожевой таймер – это аппаратная схема, позволяющая перезагрузить МК или процессора ситуации его зависания, блокировки и т.п. Устройство вызывает сброс системы, если через фиксированный момент времени содержимое определенного регистра не будет обновлено.

2) Различие в условн. переходах между AVR и 8051, case.

8051

- jz, jnz – переход если акк = 1 / ≠ 1
- jc, jnc – переход если перенос
- jb, jnb – переход если бит = 1, = 0
- jbc bit, rel – переход если бит установлен с последующ. сбросом бита
- cjne A, ad, rel — сравнение операнда 1 и 2 переход если не равно
  — // — A, #d, rel
  — // — @R, #d, rel
- Ljmp ad16 – длинный переход
- ajmp – абсолютный переход
- sjmp rel – короткий переход в границах 256 байт
- jmp @A+dptr – косвенный относит. переход

case a of
1.
2.
3.

AVR

- brbc bit, Label – переход на метку label, SREG.bit = 0
- brbs bit, label – // – .bit = 1

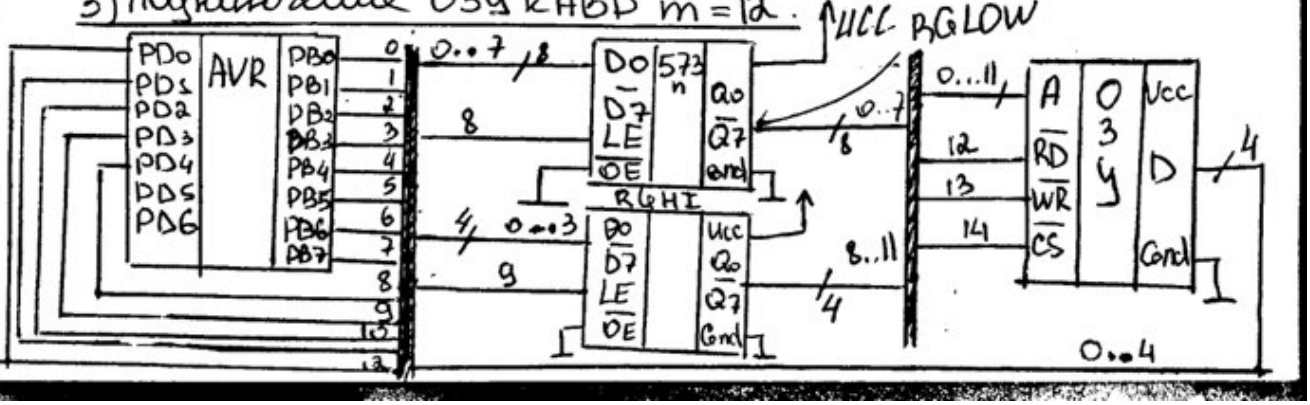
SREG – регистр состояния
I (111) – флаг разреш. прерыв.
T (110) – пользовательский флаг
H (101) – флаг полупереноса
S (100) – флаг знака 1 – "–"
V (011) – переполнение
N (010) – флаг отриц. значения
Z (001) – флаг 0
C (000) флаг переноса

Команды пропуска:

sbic RegIO, bit – пропустить след. команду если RegIO.bit = 0
sbis RegIO, bit – // – .bit = 1
sbrc Rg, bit – пропустить след. команду если Rg.bit = 0
sbrs Rg, bit – // – Rg.bit = 1
cpse Rd, Rs – // – (Rd – Rs) = 0

case rs of
r2
r3
r4

3) Подключение ОЗУ к AVR m = 12.



1) **RISC** – процессоры с сокращенной системой команд. Эти процессоры обычно имеют большой набор однородных регистров универсального назначения (РОНы), их сис. команд отличается простотой. В результате аппаратная реализация такой архитектуры позволяет выполнять инструкции за один такт синхронизации, а длины самих команд унифицированы, кроме того, в арифметических операциях операндами могут быть только регистры.

2) **Сравнить внеш. прерывания (инициализ. и подсчет пр.)**

8051 5 прерыв.
2 – по перепаду
2 – от источников
1 – UART

Регистры IE, IP IE:

7	6	5	4	3	2	1	0
EA	–	–	ES	ET_1	EX_1	ET_0	EX_0

EX_0, EX_1 – биты разреш. внеш. преры. $INT0_0$ и $INT0_1$

ET_0, ET_1 – биты разреш прерыв. от таймера

ES – биты разреш UART

EA – всех прерыв 1 – разр **0** – запр.

IP:

7	6	5	4	3	2	1	0
–	–	–	PS	PT_1	PX_1	PT_0	PX_0

PS, PT_1, PT_0, TX_1, PX – биты управ. приорит. 0 – низ 1 – выс.

INT0 – 0003h } адрес. вектора
INT1 – 0013h }

```
jmp start
    org 0003h //переход к обр. INT0
    inc R0
    reti
org 0030h //на сегмент данных
start: call init //подпрог. инициал.
m1: sjmp m1
init: mov R0,#0
ORL 0A8h,#10000001b
ret.
```

AVR

10 прерываний, каждому свой вектор, расположен в нач. памяти программы.

0001 2 внеш. запрос INT0
0002 3 внеш. запрос INT1

Вх. сигнал IR о фикс. триг.

GIFR. "1" – наличие прерыв.

GIFR

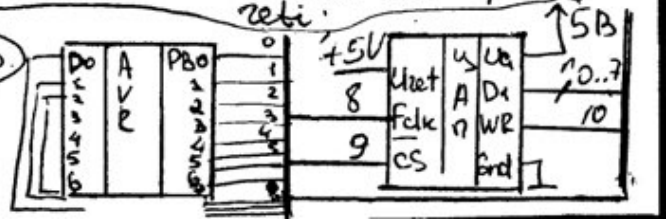
7	6
INTF1	INTF0

Регистр. маски внеш прер. GIMSk

Если прерыв разрешено битами GIMSk и флагом I_ISREG, то запрос по Вх. INT и б. выполнен, при этом соотв биты GIFR автом. сброс. при вызов. обработчиком прерывания.

```
rjmp start
    nop
rjmp count
    nop
    ...
    nop   } 8 разр.
start: LDI R16,$0
       LDI R17,$FF
       LDI R18,$80
out 7
m1: rjmp m1
count: inc R16
       out GIFR,R17
       reti
```

3) **Порты ЦАП и AVR.** (8 разр)



1) CISC – процессоры с полным набором инструкций
Состав и назначение их регистров существенно неоднородны (кол-во регистров невелико). Команды создаются для удобства, а не для эффективности исполнения. Это усложняет декодирование команд и как результат возрастает число тактов синхрониз. необходимых для выполнения команды, длины команд различны.

2) Сравнение битовых операций

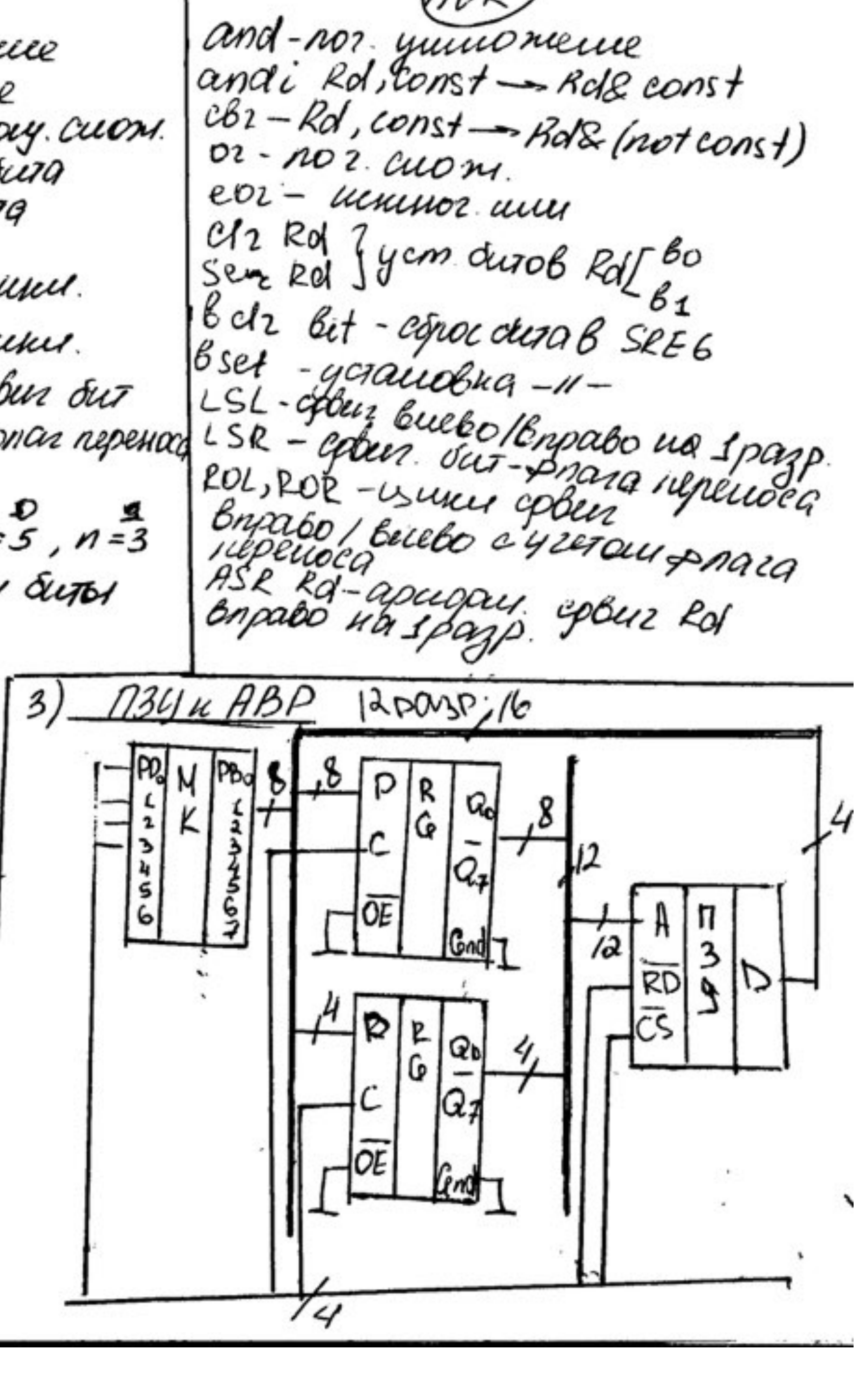
(8051)	(AVR)
anl – лог. умножение	and – лог. умножение
orl – лог. сложение	andi Rd, const → Rd& const
xrl – лог. исключающ. слож.	cbr – Rd, const → Rd& (not const)
clr – очистка, сброс бита	or – лог. слож.
setb – установка бита	eor – исключ. или
cpl – инверсия	clr Rd, ser Rd } уст. битов Rd [b0, b1]
rl – сдвиг влево цикл.	b clr bit – сброс бита в SREG
rr – сдвиг вправо цикл.	b set – установка –//–
rlc – сдвиг влево, сдвиг бит во флаг переноса	LSL – сдвиг влево/вправо на 1 разр.
rrc –//– вправо	LSR – сдвиг. бит-флага переноса
	ROL, ROR – цикл. сдвиг вправо/влево с учетом флага переноса
	ASR Rd – арифм. сдвиг Rd вправо на 1 разр.

$R_1 = 00100110$ (биты 7 6 5 4 3 2 1 0), $m=5$, $n=3$

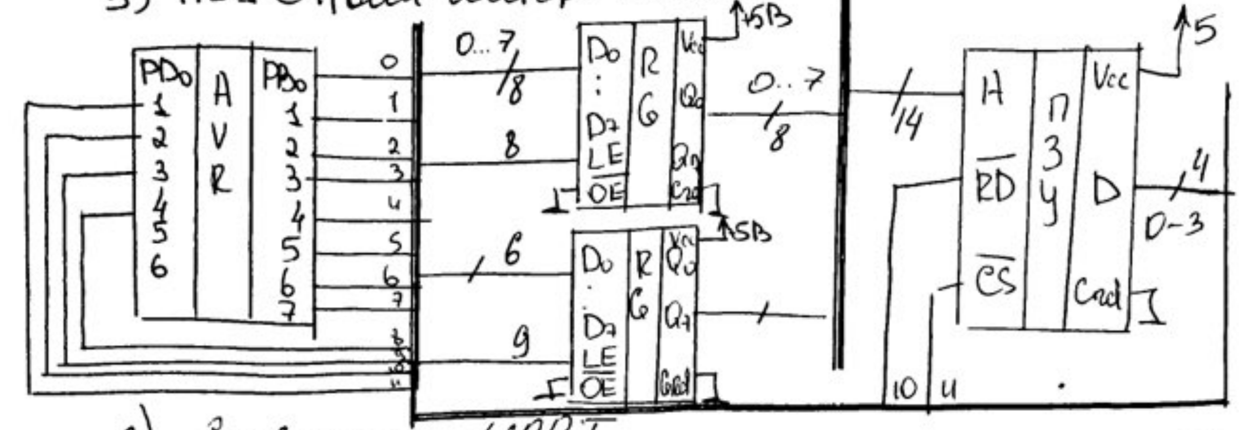
Поменять местами биты m и n

```
xseg
n: ds 1 //n
m: ds 1 //m
S: ds 1 //число
cseg
proc: mov dptr, #S
      mov a, @dptr
      mov c, acc.m
      jc m1
      mov c, acc.n
      jc exit
m1:   mov c, acc.n
      jc exit
      setb acc.n
      clr acc.m
      jmp exit
m2:   setb acc.m
      clr acc.n
exit
ret
```

3) ПЗУ к АВР 12 разр; 16



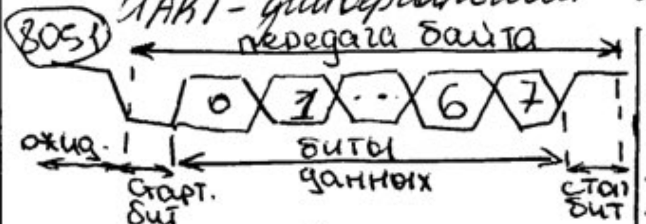
3) ПЗУ с 11 бит интер. к AVR



2) Сравнение UART

UART – универсальный асинхронный приемопередатчик

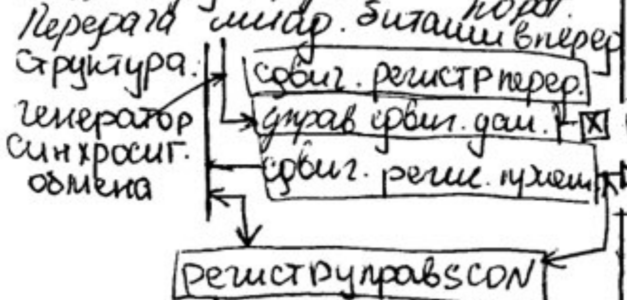
(8051)



UART – передат./прием. – послед. порт.

Передача млад. битами вперед

Структура:



↳ прерыв: регист. передатчика пуст; рег. прием заполнен

SCON – для управ раб. интерф.

0	R1	флаг. прерыв. прием.
1	T1	флаг. прерыв. передатчика
2	RB8	прием 9бита для реж 2,3
3	TB8	передача 9бита для реж 2,3
4	REN	разреш / запрет приема
5	SM2	для запр. приема сооб., в котор 9бит имеет знач. 0
6	SM1	} Режим работы UART
7	SM0	

00 – реж 0. – синхр. обмен

01 – реж 1 – асинх об, 8бит, изменяемая скорость передачи

10 – реж 2 – 8бит, фиксиров. скорость

11 – реж 3 – 9бит, изменяемая скорость передачи

MOV SCON, 01110000h – инициализ UART

- асинхр. реж раб. порта ← 01
- пров. ошиб. кадра ← 1
- разреш раб. прием ← 1
- 9бит перед. T8
- 9бит прием. R8
- флаг перед. T1
- флаг. прием. R1

(AVR)

Обеспечивает полный дуплекс при послед. обмене данных

Возможности: – данные 8 или 9 бит

- встроенная фиксация шумов
- обнаружение перегрузок
- обнаружение ошибки кадра
- обнаружение ложного старт бита

Функционирование начинается после записи в регистр передачи UDR данных, подлежащих передачи. Далее эти данные передаются в сдвиговый регистр.

UART обслуживается 4я регистр:

UDR – рег. данных устройства

USR – статусный рег. уст-ва

UCR – рег. управ. уст-ва

UBRR – рег. управ. частотой приемо-перед.

Инициализац:

```
ldi R16, low
out UBRRL, R16
ldi R16, high
out UBRRH, R16
```

```
mov SCON, #01010000h - режим 1 асинхр. 8бит
anl TMOD, #00001111h -
orl TMOD, #00100000h
mov TH0, #fdh
mov TL0, #fdh
setb TR1
jnb RI, $ - ∞ цикл ожид.
```