8. Контрольная работа

8.1. Цель контрольной работы - освоение основных принципов построения микропроцессоров путем подготовки программы на языке Ассемблер (на примере микроконтроллера семейства AVR ATmega 128).

8.2. Общие сведения о микроконтроллере ATmega 128

ATmega128 – маломощный 8-разрядный КМОП микроконтроллер, основанный на расширенной AVR RISC-архитектуре. Высокие характеристики семейства **AVR** обеспечиваются следующими особенностями архитектуры:

- В качестве памяти программ используется внутренняя flash-память. Она организована в виде матрицы 16-разрядных ячеек и может загружаться программатором, либо через порт SPI;
- Система команд включает 133 инструкций;
- 16-разрядные память программ и шина команд вместе с одноуровневым конвейером позволяют выполнить большинство инструкций за один такт синхрогенератора (50 нс при частоте F_{OSC}=20 МГц);
- Память данных имеет 8-разрядную организацию. Младшие 32 адреса пространства занимают регистры общего назначения, далее следуют 64 адреса регистров ввода-вывода, затем внутреннее ОЗУ данных объемом до 4096 ячеек. Возможно применение внешнего ОЗУ данных объемом до 60 Кбайт;
- Внутренняя энергонезависимая память типа EEPROM объемом до 4 Кбайт представляет собой самостоятельную матрицу, обращение к которой осуществляется через специальные регистры ввода-вывода.

Регистровый файл с быстрым доступом содержит 32 регистра общего назначения (РОН), которые включены в сквозное адресное пространство ОЗУ данных и занимают младшие адреса (рис. 1). РОНы предназначены для хранения адресов и данных. Файл регистров общего назначения прямо связан с арифметико-логическим устройством (АЛУ), каждый из регистров способен работать как аккумулятор. Шесть регистров из 32 могут использоваться как три 16-разррядных регистра косвенного адреса для эффективной адресации в пределах памяти данных. Данные 16-разрядные регистры называются X-регистр, Y-регистр и Z-регистр.

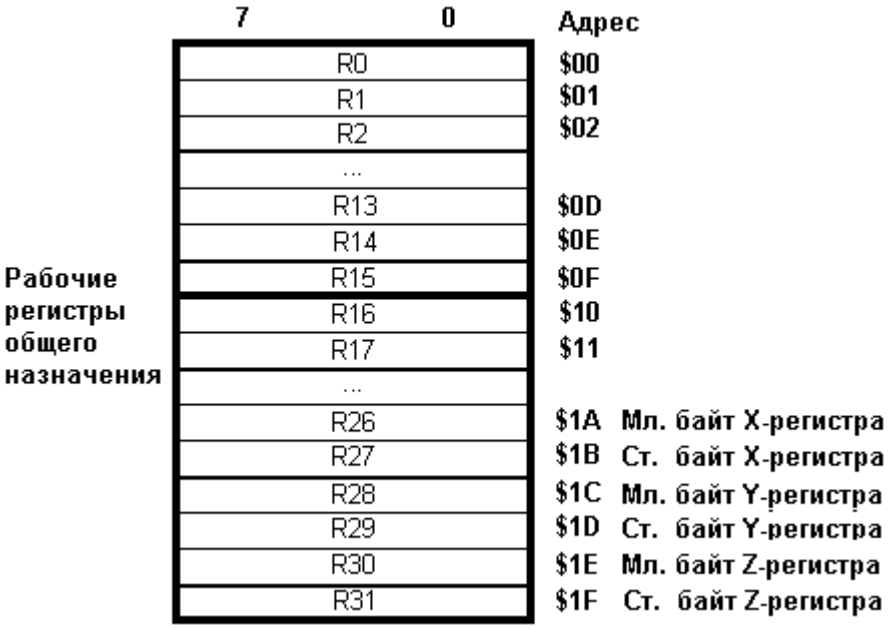

Рис. 1. Регистры общего назначения микроконтроллера ATmega128

Микроконтроллеры AVR оснащены 16-разрядным указателем стека SP, размещенным в двух регистрах ввода-вывода (рис. 2) – для хранения старших разрядов (регистр SPH) и младших (регистр SPL). Поскольку микроконтроллеры ATmega128 поддерживают объем ОЗУ до 64 Кбайт, то используются все 16 разрядов указателя стека.

Указатель стека указывает на область в ОЗУ данных, в которой размещается стек подпрограмм и прерываний, который обычно используется для хранения временных данных, для хранения локальных переменных и для хранения адресов возврата при прерываниях и вызовах подпрограмм. Регистр указателя стека указывает на вершину стека. Начальный адрес указателя должен задаваться программно перед вызовом подпрограмм и разрешением прерываний. Начальное значение должно быть больше $60. Указатель стека декрементируется (уменьшается) на единицу при каждом занесении командой PUSH данных в стек, и на две единицы при занесении в стек адреса при вызове подпрограммы или процедуры прерывания.

Указатель стека инкрементируется (увеличивается) на единицу при извлечении данных из стека командой POP, и на две единицы при извлечении адреса из стека при возврате из подпрограммы (RET) или возврате из процедуры прерывания (RETI).

Команды микроконтроллера ATmega128

При создании программы для микроконтроллера на языке Ассемблер разработчик оперирует программно доступными ресурсами микропроцессорной системы. У микроконтроллера ATmega128 эти ресурсы

включают: программно доступные регистры микроконтроллера, внутреннюю память данных, внешнюю память данных.

Каждая команда языка Ассемблер сообщает процессору выполняемую операцию и методы доступа к операндам (данными с которыми производятся операции). Командная строка Ассемблера включает метку (символический адрес), мнемонику (символическое имя) команды, поле операндов, комментарий. Имя команды однозначно связано с выполняемой ею операцией.

Методы адресации представляют собой набор механизмов доступа к операндам. При прямой адресации для пересылки данных из ОЗУ в регистры общего назначения и обратно используются команды LDS и STS соответственно, при этом адрес ячейки ОЗУ указывается в команде. При косвенной адресации используются команды LD и ST, в этом случае адрес ячейки ОЗУ хранится в регистровых парах X, Y или Z (рис. 1). Команда LDI загружает регистр общего назначения непосредственно константой (только регистры R16 – R32). Основные команды микроконтроллера ATmega128, необходимые для выполнения контрольной работы, приведены в таблице 1.

Основные команды микроконтроллера ATmega 128 Таблица 1

Мнемо-ника	**Операн-ды**	**Описание**	**Операция**	**Флаги**	**Кол-во циклов**
1	2	3	4	5	6
MOV	Rd,Rr 0≤d≤31 0≤r≤31	Копировать регистр	Rd ← Rr	Нет	1
LDI	Rd,K 16≤d≤31 0≤k≤255	Загрузить непосредственное значение	Rd ← K	Нет	1
LDS	Rd,k 0≤d≤31 0≤k≤65535	Загрузить из ОЗУ	Rd ← (k)	Нет	3
LD	Rd,X Rd,Y Rd,Z 0≤d≤31	Загрузить косвенно	Rd ← (X) Rd ← (Y) Rd ← (Z)	Нет	2
STS	k,Rr 0≤r≤31 0≤k≤65535	Загрузить непосредственно в ОЗУ	(k) ← Rr	Нет	3
ST	X,Rr Y,Rr Z,Rr	Записать косвенно	(X) ← Rr (Y) ← Rr (Z) ← Rr	Нет	2

	0≤r≤31				
LPM		Загрузить байт из памяти программ	R0 ← (Z)	Нет	3
OUT	P,Rr 0≤r≤31 0≤P≤63	Записать данные из регистра в порт I/O	P ← Rr	Нет	1
PUSH	Rr 0≤r≤31	Сохранить регистр в стеке	STACK ← Rr	Нет	2
POP	Rd 0≤d≤31	Выгрузить регистр из стека	Rd ← STACK	Нет	2

Таблица 1 (продолжение)

1	2	3	4	5	6
ADD	Rd,Rr 0≤d≤31 0≤r≤31	Сложить без переноса	Rd ← Rd + Rr	Z, C, N, V, H	1
ADC	Rd,Rr 0≤d≤31 0≤r≤31	Сложить с переносом	Rd ← Rd + Rr + C	Z, C, N, V, H	1
SUB	Rd,Rr 0≤d≤31 0≤r≤31	Вычесть без заема	Rd ← Rd - Rr	Z, C, N, V, H	1
SBC	Rd, Rr 0≤d≤31 0≤r≤31	Вычесть с заемом	Rd ← Rd - Rr - C	Z, C, N, V, H	1
INC	Rd 0≤d≤31	Инкрементировать	Rd ← Rd + 1	Z, N, V	1
DEC	Rd 0≤d≤31	Декрементировать	Rd ← Rd - 1	Z, N, V	1
CPSE	Rd,Rr 0≤d≤31, 0≤r≤31	Сравнить и пропустить, если равно	if Rd=Rr then PC ← PC + 2 (or 3)	Нет	1/2/3
RJMP	k 2K≤k≤2K	Перейти относительно	PC ← PC + k + 1	Нет	2
CALL	K 0≤k≤64K	Выполнить длинный вызов подпрограммы	PC ← k	Нет	4
RET		Вернуться из подпрограммы	PC ← STACK	Нет	4

8.3. Задание на контрольную работу

Написать программу на языке Ассемблер микроконтроллера AVR ATmega128, выполняющую задачу, в соответствии с вариантом. Вариант задания выбирается по таблицам 2 и 3 следующим образом. По предпоследней цифре студенческого билета в таблице 2 выбирается номер варианта задания (текст приведен ниже), виды адресации при пересылке данных (для случаев, если вид адресации при конкретной пересылке не указан в задании), значение указателя стека SP. По последней цифре студенческого билета в таблице 3 выбираются значения операндов A1…A6 и номера ячеек ОЗУ M1…M4. Все числа – шестнадцатеричные.

Например, последние цифры студенческого билета 45. По таблице 2 в первом столбце находим №4 (предпоследняя цифра студенческого билета) и выбираем данные для этой строчки: вариант задания 1 (столбец 2); виды адресации (если не указаны по тексту задания): при пересылке данных из ОЗУ в РОНы адресация прямая (столбец 3), при пересылке данных из РОНов в ОЗУ – косвенная (столбец 4); значение указателя стека 10F5 (столбец 5). По таблице 3 аналогично находим в первом столбце №5 (последняя цифра студенческого билета) и выбираем значения операндов (A1...A6) и номера ячеек ОЗУ (M1...M4): A1=10 (столбец 2), A2=33 (столбец 3), A3=99 (столбец 4), A4=57 (столбец 5), A5=56 (столбец 6), A6=33 (столбец 7), M1=0058 (столбец 8), M2=0059 (столбец 9), M3=0060 (столбец 10), M4=0061 (столбец 11).

Отчет должен содержать:
– задание на контрольную работу;
– выбор и описание используемых команд ассемблера;
– текст программы с подробными комментариями.

Исходные данные Таблица 2

№	Номер варианта задания	Адресация при пересылке данных из ОЗУ в РОНы	Адресация при пересылке данных из РОНов в ОЗУ	SP
1	*2*	*3*	*4*	*5*
1	1	прямая	косвенная	10F1
2	2	косвенная	прямая	10F2
3	1	прямая	прямая	10F3
4	2	косвенная	косвенная	10F4
5	1	прямая	косвенная	10F5
6	2	косвенная	прямая	10F6
7	1	прямая	прямая	10F7

8	2	косвенная	косвенная	10F9
9	1	прямая	косвенная	10FA
0	2	косвенная	прямая	10FB

Исходные данные (продолжение) Таблица 3

№	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**M1**	**M2**	**M3**	**M4**
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
1	17	5	67	76	65	12	0041	0043	0044	0045
2	2A	67	54	45	45	2B	0046	0047	0048	0049
3	37	98	34	44	78	34	0050	0051	0052	0053
4	12	45	67	45	34	65	0054	0055	0056	0057
5	10	33	99	57	56	33	0058	0059	0060	0061
6	44	45	65	21	33	22	0062	0063	0064	0065
7	54	67	34	46	23	87	0066	0067	0068	0069
8	32	5A	56	78	44	98	0070	0071	0072	0073
9	23	4	87	43	90	43	0074	0075	0076	0077
0	4C	32	44	23	54	12	0078	0079	0080	0081

Варианты заданий

Вариант 1.

Сложить три числа A1, A2, A3 (без учета единица переноса). Результат записать в ячейку ОЗУ M1.

Загрузить ячейки ОЗУ M2 и M3 числами A4 и A5 соответственно, используя косвенную адресацию (регистровые пары X и Y).

Записать в регистр R16 число A6.

Осуществить вызов подпрограммы, в которой требуется:

– сохранить указатели X, Y и содержимое регистра R16 в стеке. Начальный адрес стека SP;

– записать в регистр R16 содержимое ячейки ОЗУ M4 и сравнить с содержимым ячейки ОЗУ M2, в случае неравенства увеличить содержимое регистра R16 на единицу;

– извлечь сохраненные данные из стека, поменяв содержимое указательных регистров X и Y местами.

Вариант 2.

Сложить два числа A1 и A2 (без учета единица переноса), результат уменьшить на единицу и записать в ячейку ОЗУ M1.

Загрузить ячейки ОЗУ M2 и M3 числами A3 и A4 соответственно, используя косвенную адресацию (регистровые пары Y и Z).

Записать в регистр R16 число A5.

Осуществить вызов подпрограммы, в которой требуется:

– сохранить указатели Y, Z и содержимое регистра R16 в стеке. Начальный адрес стека SP;

– записать в регистр R16 содержимое ячейки ОЗУ M4 и сравнить с содержимым ячейки ОЗУ M2, в случае неравенства прибавить к содержимому регистра R16 число A6;

– извлечь сохраненные данные из стека, поменяв содержимое указательных регистров X и Y местами.

Пример выполнения задания

Задание.

Сложить три числа 16, 72, 35 (без учета единица переноса). Результат записать в ячейку ОЗУ 0075.

Загрузить ячейки ОЗУ 0078 и 0080 числами 2A и 55 соответственно, используя косвенную адресацию (регистровые пары X и Y).

Записать в регистр R16 число 15.

Осуществить вызов подпрограммы, в которой требуется:

– сохранить указатели X, Y и содержимое регистра R16 а стеке. Начальный адрес стека 10FF;

– записать в регистр R16 содержимое ячейки ОЗУ 0090 и сравнить с содержимым ячейки ОЗУ 0078, в случае неравенства увеличить содержимое регистра R16 на единицу;

– извлечь сохраненные данные из стека, поменяв содержимое указательных регистров X и Y местами.

При пересылке данных из ОЗУ в РОНы использовать косвенную адресацию, при пересылке данных из РОНов в ОЗУ использовать прямую адресацию.

Выбор команд для программы.

В начале программы необходимо задать адрес стека, где будут сохранены необходимые данные при вызове подпрограммы. Так как указатель стека реализован в виде двух регистров ввода вывода, то для записи в данные регистры необходимо использовать команду OUT:

– для записи в регистр SPL:

OUT SPL,Rr ;

– для записи в регистр SPH:

OUT SPH,Rm ,

где m – номер регистра, из которого пересылаются данные в стек (r=0…31).

Для записи операндов в РОНы необходимо использовать команду

LDI Rd, k ,
где d – номер регистра в который записывается операнд (d=16…31);
k – операнд.
Для пересылки данных из ОЗУ в РОНы при косвенной адресации используются команды
LD Rd, X ;
LD Rd, Y ;
LD Rd, Z ,
где X, Y, Z – регистры для хранения адреса ячейки ОЗУ из которой пересылаются данные. Каждый из этих регистров образует пара регистров в которых хранится младшие и старшие 8 разрядов адреса соответственно. Регистр X образуют регистры R26 (для хранения младших разрядов), R27 (для хранения старших разрядов). Аналогично регистр Y образуют регистры R28 и R29, регистр Z образуют регистры R30, R31. Перед использованием команды LD адрес должен быть записан в соответствующие регистры, которые используются в команде.

В случае прямой адресации при пересылке данных из ОЗУ в РОНы используется команда
LDS Rm, n ,
где n – адрес ячейки ОЗУ.
Для пересылки данных из РОНов в ОЗУ при прямой адресации используется команда
SDS Rm, n .

В случае косвенной адресации адрес ячейки ОЗУ хранится в одном из регистров X, Y или Z и для записи данных в ОЗУ используются команды:
LD X, Rd ;
LD Y, Rd ;
LD Z, Rd .

Для выполнения операции сложения используем команду
ADD Ra, Rb ,
где a, b – номера регистров, где хранятся операнды. Результат будет записан в регистр Ra.

Для выполнения требуемой в задании операции сравнения используем команду
CPSE Ra, Rb .
Данная команда сравнивает содержимое регистров Ra, Rb и в случае равенства пропускает следующую команду.

Для увеличения содержимого регистра на единицу можно использовать команду
INC Rm.

Для уменьшения содержимого регистра на единицу можно использовать команду
DEC Rm.

Для вызова подпрограммы используем команду CALL.

В подпрограмме при сохранении содержимого регистра Rm стеке используем команду
PUSH Rm .
Для извлечения из стека в регистр Rm используется команда
POP Rm .
При этом извлечение данных из стека происходит в обратном порядке, т.е. при первой команде POP Rm в регистр Rm помещаются данные, записанные последней командой PUSH, второй командой POP Rm в регистр Rm извлекаются из стека данные, записанные предпоследней командой POP и т.д. Учитываю данную особенность извлечения данных из стека можно реализовать требуемую в задании перемену содержимого регистров, которое записывается в стек.

Для выполнения переходов в программе можно использовать команду RJMP. Данную команду можно использовать в конце программы при «зацикливании» программы, чтобы не осуществлялся переход к следующей команде.

Текст программы.

```
; Инициализация указателя стека
; Адрес стека 10FF, для его задания в регистр SHL помещаем младший байт
FF, ; а в регистр SPH – старший 10.
LDI  R20, $FF ; загрузка регистра R20 младшим байтом адреса начала стека
OUT  SPL, R20 ; загрузка младшего байта указателя стека
LDI  R20, $10 ; загрузка регистра R20 старшим байтом адреса начала стека
OUT  SPH, R20 ; загрузка младшего байта указателя стека

; Выполнение сложения трех чисел и записи результата в ОЗУ
LDI R17, $16 ; загрузка регистра R17 числом 16
LDI R18, $72 ; загрузка регистра R18 числом 72
LDI R19, $35 ; загрузка регистра R19 числом 35
ADD R17, R18 ; суммирование содержимого регистров R17 и R18
ADD R17,R19 ; суммирование содержимого регистров R17 и R19
; загрузка в регистровую пару Z адреса ячейки ОЗУ 0075
LDI R30, $75
LDI R31, $00
ST Z, R17; загрузка ячейки ОЗУ с адресом 0075 значением из регистра R17
```

```
; загрузка ячеек ОЗУ 0078 и 0080 числами 2A и 55 соответственно
LDI R20, $2A ; загрузка регистра R20 числом 2A
LDI R21, $55 ; загрузка регистра R21 числом 55
; загрузка в регистровую пару X адреса ячейки ОЗУ 0078
LDI R26, $78
LDI R27, $00
; загрузка в регистровую пару Y адреса ячейки ОЗУ 0080
LDI R28, $80
LDI R29, $00
ST X, R20 ; загрузка ячейки ОЗУ с адресом 0078 значением из регистра R20
ST Y, R21 ; загрузка ячейки ОЗУ с адресом 0075 значением из регистра R21

LDI R16, $15 ; загрузка регистра R16 числом 15

CALL ROUT ; вызов подпрограммы ROUT
; зацикливание программы
LOOP:
RJMP LOOP

; ПОДПРОГРАММА
ROUT:
; сохранение указателя X в стеке
PUSH R26
PUSH R27
; сохранение указателя Y в стеке
PUSH R28
PUSH R29
LDS R16, $0090 ; загрузка регистра R16 содержимым ячейки ОЗУ с адресом
0090
LDS R17, $0078 ; загрузка регистра R17 содержимым ячейки ОЗУ с адресом
0078

CPSE R16, R17 ; пропустить следующую команду если значения регистров
R16 и R17 равны
INC R16 ; увеличить содержимое регистра R16 на единицу

; извлечение сохраненных данных из стека и замена содержимого
указательных
; регистров
; извлечение сохраненного содержимого Y в X
POP R27
POP R26
; извлечение сохраненного содержимого X в Y
POP R29
```

```
POP R28
RET ; возврат из подпрограммы.
```