МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ»

Факультет информатики и
вычислительной техники

Кафедра ИСП

Отчет
по лабораторной работе № 2

по дисциплине «Машинно-зависимые языки программирования»
Вариант 4

Выполнил: студент группы ПС-11
Щеглов Г.С
Проверил: Баев А.А.

г. Йошкар-Ола
2024

Цель работы: Научиться восстанавливать ассемблерный код по HEX-коду, построить алгоритм и написать код на C.

Задания на лабораторную работу:

1. Восстановить HEX код в ассемблерный код
2. Построить алгоритм и написать код на C

1. Теоретические сведения

Пример решения:
:100000000C9434000C943E000C943E000C943E0082

:10 0000 00 0C94 3400 0C94 3E00 0C94 3E00 0C94 3E00 82
:NN AAAA CC

Поле NN определяет количество байтов данных в строке (в нашем случае 16 байтов). Поле AAAA – это начальный адрес, с которого данные будут записаны в память микроконтроллера.

За адресом следует поле команды CC. Программатор, ориентируясь на поле CC, распознает функциональное назначения строки. Ассемблер и другие компиляторы языков высокого уровня для AVR могут установить следующие значения данного параметра:
00 – в строке находятся данные для записи в память,
01 – последняя строка в файле,
02 – строка содержит начальный адрес сегмента памяти,
04 – строка содержит адрес в пределах сегмента памяти.

В данной строке CC=00 (т.е. строка предназначена для записи данных). За полем CC (кроме команды 01) идут непосредственно данные в количестве, определяемом параметром NN. Последнее поле SS – контрольная сумма. Сумма всех байтов в неповрежденной строке без учета переполнения всегда нулевая

Меняем байты местами
:10 0000 00 940C 0034 940C 003E 940C 003E 940C 003E 82

Если начинается на 94, то это команды jmp или call – они занимают 4 байта

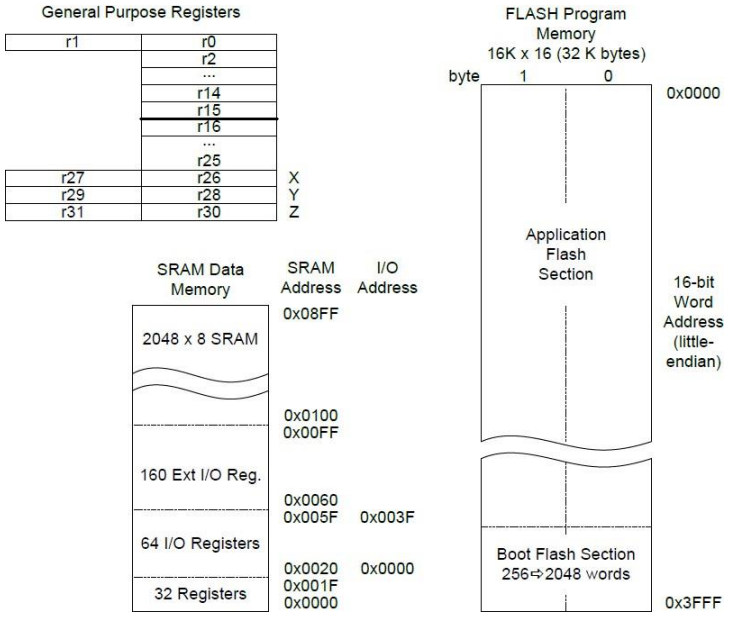
32-разрядный код операции JMP:

1001	010k	kkkk	110k
kkkk	kkkk	kkkk	kkkk

940C0034
1001 010 0 0000 110 00000000000110100

Таким образом, k = 0 0 0000 00000000 00110100, однако при дизассемблировании нужно для команд rjmp, jmp, call сместить адрес на один бит влево, тогда k = 0110 1000 = 0x68

Почему необходимо выполнять смещение адреса? Смещение влево увеличиваем значение в два раза. В коде команды содержится информация о значении программного счетчика, а память команд (FLASH program Memory) у нас 16ти битная и адреса указываются соответственно. Прошивка и наш код хранятся в нашей системе с побайтовыми адресами. Таким образом, чтобы нам легче было ориентироваться в коде и файле прошивки, мы смещение указываем в байтах.



В остальных случаях коды 16ти битные, так в строке:

:10 0060 00 0C943E00 0C943E00 1124 1FBE CFEF D8E0 4C

11 24 поменяем местами 24 11

16-разрядный код операции:

0010	01rd	dddd	rrrr

0010 0100 0001 0001

r = 00001 d = 00001

68: 11 24 eor r1, r1

С остальными строками производим те же операции, однако от зависимости команды производим дополнительные действия.

Дополнительные действия для ldi, subi:

1. Регистр d переводим в 2- ую СС, а k в 16-ую.

2. Так как регистр d начинает свой отсчёт от 16, а k от 0, следовательно к d нужно прибавить 16, а k остается так же.

Дополнительные действия для rjmp, breq, bren:

1. Обращаем на первую цифру маски.
2. Если она начинается на ‘1’, то результат будет с минусом и нужно выполнить следующие операции:
- Строки инверсируем, к получившейся строке +1 и сделать сдиг на 1 бит влево
- Получившееся число переводим в 10-ую систему счисления и перед ним вставим ‘-’
3. Если она начинается на ‘0’, то результат будет с плюсом и нужно выполнить следующие операции:
- Получившееся k смещаем на 1 бит влево и переводим в 10 СС

Нумерация строк идёт в 16-ой СС и номер зависит от того какая команда была до этого:
- Если была call или jmp, то прибавляется 4 байта, то есть: старая строчка + 4 = новая строчка
- Если это любая другая команда, то они занимают по 2 байта, то есть: старая строчка + 2 = новая строчка

2. Практическая часть

Исходный hex-код

```
:100000000C9434000C943E000C943E000C943E0082
:100010000C943E000C943E000C943E000C943E0068
:100020000C943E000C943E000C943E000C943E0058
:100030000C943E000C943E000C943E000C943E0048
:100040000C943E000C943E000C943E000C943E0038
:100050000C943E000C943E000C943E000C943E0028
:100060000C943E000C943E0011241FBECFEFD8E04C
:10007000DEBFCDBF0E9440000C9450000C940000E5
:10008000389A3D98459A359902C0409A01C04098E7
:1000900028E880EF91E2215080409040E1F7F3CFD3
:0400A000F894FFCF02
:00000001FF
```

Задача 1.

Вариант 4. Результаты:

```
0: 0C 94 34 00 jmp 0x68
4: 0C 94 3E 00 jmp 0x7C
8: 0C 94 3E 00 jmp 0x7C
c: 0C 94 3E 00 jmp 0x7C
10: 0C 94 3E 00 jmp 0x7C
14: 0C 94 3E 00 jmp 0x7C
18: 0C 94 3E 00 jmp 0x7C
1c: 0C 94 3E 00 jmp 0x7C
20: 0C 94 3E 00 jmp 0x7C
24: 0C 94 3E 00 jmp 0x7C
28: 0C 94 3E 00 jmp 0x7C
2c: 0C 94 3E 00 jmp 0x7C
30: 0C 94 3E 00 jmp 0x7C
34: 0C 94 3E 00 jmp 0x7C
```

38: 0C 94 3E 00 jmp 0x7C
3c: 0C 94 3E 00 jmp 0x7C
40: 0C 94 3E 00 jmp 0x7C
44: 0C 94 3E 00 jmp 0x7C
48: 0C 94 3E 00 jmp 0x7C
4c: 0C 94 3E 00 jmp 0x7C
50: 0C 94 3E 00 jmp 0x7C
54: 0C 94 3E 00 jmp 0x7C
58: 0C 94 3E 00 jmp 0x7C
5c: 0C 94 3E 00 jmp 0x7C
60: 0C 94 3E 00 jmp 0x7C
64: 0C 94 3E 00 jmp 0x7C
68: 11 24 eor r1 r1
6a: 1F BE out 0x3F, r1
6c: CF EF ldi r28, 0xFF
6e: D8 E0 ldi r29, 0x8
70: DE BF out 0x3E, r29
72: CD BF out 0x3D, r28
74: 0E 94 40 00 call 0x80
78: 0C 94 50 00 jmp 0xA0
7c: 0C 94 00 00 jmp 0
80: 38 9A sbi 0x7, 0
82: 3D 98 cbi 0x7, 5
84: 45 9A sbi 0x8, 5
86: 35 99 sbic 0x6, 5
88: 02 C0 rjmp +4
8a: 40 9A sbi 0x8, 0
8c: 01 C0 rjmp +2
8e: 40 98 cbi 0x8, 0
90: 28 E8 ldi r18, 0x88
92: 80 EF ldi r24, 0xF0
94: 91 E2 ldi r25, 0x21
96: 21 50 subi r18, 0x01
98: 80 40 sbci r24, 0x0
9a: 90 40 sbci r25, 0x0
9c: E1 F7 brne -8
9e: F3 CF rjmp -26
a0: F8 94 cli
a2: FF CF rjmp -2

Ход решения:

1.

:10 0000 00 0C94 3400 0C94 3E00 0C94 3E00 0C94 3E00 82

940C0034 940C003E 940C003E 940C003E

1.1
940C0034
1001 0100 0000 1100 0000 0000 0011 0100 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 0100
k = 0110 1000
0x68
Получаем: 0C 94 34 00 jmp 0x68

1.2
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

1.3
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

1.4
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

2.
:10 0010 00 0C94 3E00 0C94 3E00 0C94 3E00 0C94 3E00 68
940C003E 940C003E 940C003E 940C003E

2.1
940C003E

1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

2.2
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

2.3
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

2.4
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

3.
:10 0020 00 0C94 3E00 0C94 3E00 0C94 3E00 0C94 3E00 58
940C003E 940C003E 940C003E 940C003E

3.1
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk

k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

3.2
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

3.3
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

3.4
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

4.
:10 0030 00 0C94 3E00 0C94 3E00 0C94 3E00 0C94 3E00 48
940C003E 940C003E 940C003E 940C003E

4.1
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C

Получаем: 0C 94 3E 00 jmp 0x7C

4.2
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

4.3
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

4.4
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

5.
:10 0040 00 0C94 3E00 0C94 3E00 0C94 3E00 0C94 3E00 38
940C003E 940C003E 940C003E 940C003E

5.1
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

5.2

940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

5.3
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

5.4
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

6.
:10 0050 00 0C94 3E00 0C94 3E00 0C94 3E00 0C94 3E00 28
940C003E 940C003E 940C003E 940C003E

6.1
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

6.2
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk

k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

6.3
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

6.4
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

7.
:10 0060 00 0C94 3E00 0C94 3E00 1124 1FBE CFEF D8E0 4C
940C003E 940C003E 2411 BE1F EFCF E0D8

7.1
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C
Получаем: 0C 94 3E 00 jmp 0x7C

7.2
940C003E
1001 0100 0000 1100 0000 0000 0011 1110 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0011 1110
k = 0111 1100
0x7C

Получаем: 0C 94 3E 00 jmp 0x7C

7.3
2411
0010 0100 0001 0001 eor
0010 01rd dddd rrrr
r = 00001 d = 00001
r = r1 d = r1
Получаем: 11 24 eor r1, r1

7.4
BE1F
1011 1110 0001 1111 out
1011 1PPr rrrr PPPP
P = 111111 r = 00001
P = 0x3F r = r1
Получаем: 1F BE out 0x3F, r1

7.5
EFCF
1110 1111 1100 1111 ldi
1110 KKKK dddd KKKK
K = 11111111 d = 1100
K = 0xFF d = r(12+16) = r28
Получаем: CF EF ldi r28, 0xFF

7.6
E0D8
1110 0000 1101 1000 ldi
1110 KKKK dddd KKKK
K = 00001000 d = 1101
K = 0x8 d = r(13+16) = r29
Получаем: D8 E0 ldi r29, 0x8

8.
:10 0070 00 DEBF CDBF 0E94 4000 0C94 5000 0C94 0000 E5
BFDE BFCD 940E0040 940C0050 940C0000

8.1
BFDE
1011 1111 1101 1110 out
1011 1PPr rrrr PPPP
P = 111110 r = 11101

P = 0x3E r = r(13+16) = r29
Получаем: DE BF 0x3E, r29

8.2
BFCD
1011 1111 1100 1101 out
1011 1PPr rrrr PPPP
P = 111101 r = 11100
P = 0x3D r = r(12+16) = r28
Получаем: CD BF 0x3D, r28

8.3
940E0040
1001 0100 0000 1110 0000 0000 0100 0000 call
1001 010k kkkk 111kkkkk kkkk kkkk kkkk
k = 0100 0000
k = 1000 0000
0x80
Получаем: 0E 94 40 00 call 0x80

8.4
940C0050
1001 0100 0000 1100 0000 0000 0101 0000 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0101 0000
k = 1010 0000
0xA0
Получаем: 0C 94 50 00 jmp 0xA0

8.5
940C0000
1001 0100 0000 1100 0000 0000 0000 0000 jmp
1001 010k kkkk 110kkkkk kkkk kkkk kkkk
k = 0000 0000
k = 0
Получаем: 0C 94 00 00 jmp 0

9.
:10 0080 00 389A 3D98 459A 3599 02C0 409A 01C0 4098 E7
9A38 983D 9A45 9935 C002 9A40 C001 9840

9.1
9A38
1001 1010 0011 1000 sbi
1001 1010 PPPP Pbbb
P = 00111 b =000
P = 0x7 b = 0
Получаем: 38 9A sbi 0x7, 0

9.2
983D
1001 1000 0011 1101 cbi
1001 1000 PPPP Pbbb
P = 00111 b = 101
P = 0x7 b = 5
Получаем: 3D 98 cbi 0x7, 5

9.3
9A45
1001 1010 0100 0101 sbi
1001 1010 PPPP Pbbb
P = 01000 b = 101
P = 0x8 b = 5
Получаем: 45 9A sbi 0x8, 5

9.4
9935
1001 1001 0011 0101 sbic
1001 1001 PPPP Pbbb
P = 00110 b = 101
P = 0x6 b = 5
Получаем: 35 99 sbic 0x6, 5

9.5
C002
1100 0000 0000 0010 rjmp
1100 kkkk kkkk kkkk
k = 0000 0000 0010

k = 0000 0000 0100
k = 4
Получаем 02 C0 rjmp +4

9.6
9A40
1001 1010 0100 0000 sbi
1001 1010 PPPP Pbbb
P = 01000 b = 000
P = 0x8 b = 0
Получаем: 40 9A sbi 0x8, 0

9.7
C001
1100 0000 0000 0001 rjmp
1100 kkkk kkkk kkkk
k = 0000 0000 0001
k = 0000 0000 0010
k = 2
Получаем: 01 C0 rjmp +2

9.7
9840
1001 1000 0100 0000 cbi
1001 1000 PPPP Pbbb
P = 01000 b = 000
P = 0x8 b=0
Получаем: 40 98 cbi 0x8, 0

10.
:10 0090 00 28E8 80EF 91E2 2150 8040 9040 E1F7 F3CF D3
E828 EF80 E291 5021 4080 4090 F7E1 CFF3

10.1
E828
1110 1000 0010 1000 ldi
1110 KKKK dddd KKKK
K = 10001000 d = 0010
K = 0x88 d = r(2+16) = r18
Получаем: 28 E8 ldi r18, 0x88

10.2
EF80

1110 1111 1000 0000 ldi
1110 KKKK dddd KKKK
K = 11110000 d = 1000
K = 0xF0 d = r(8+16)
Получаем: 80 EF ldi r24, 0xF0

10.3
E291
1110 0010 1001 0001
1110 KKKK dddd KKKK ldi
K = 00100001 d = 1001
K = 0x21 d = r(9+16)
Получаем: 91 E2 ldi r25, 0x21

10.4
5021
0101 0000 0010 0001 subi
0101 KKKK dddd KKKK
K = 00000001 d = 0010
K = 0x1 d = r(2+16)= r18
Получаем: 21 50 subi r18, 0x1

10.5
4080
0100 0000 1000 0000 sbci
0100 KKKK dddd KKKK
k = 00000000 d = 1000
k = 0x0 d = r(8+16) = r24

Получаем: 80 40 sbci r24, 0x0

10.6
4090
0100 0000 1001 0000 sbci
0100 KKKK dddd KKKK
K = 00000000 d = 1001
K = 0x0, d = r(9+16) = r25
Получаем: 90 40 sbci r25, 0x0

10.7
F7E1
1111 0111 1110 0001 brne
1111 01kk kkkk k001

k = 1111100
k = 0000011 + 1
k = 0000100
k = 0001000
k = -8
Получаем: E1 F7 brne -8

10.8
CFF3
1100 1111 1111 0011 rjmp
1100 kkkk kkkk kkkk
k = 111111110011
k = 000000001100 + 1
k = 000000001101
k = 000000011010
k = -26
Получаем: F3 CF rjmp -26

11.
:04 00A0 00 F894 FFCF 02
94F8 CFFF

11.1
94F8
1001 0100 1111 1000 cli
Получаем: F894 cli

11.2
CFFF
1100 1111 1111 1111 rjmp
1100 kkkk kkkk kkkk
k = 1111 1111 1111
k = 0000 0000 0000 + 1
k = 0000 0000 0001
k = 0000 0000 0010
k = 2
Получаем: FF CF rjmp -2

Задача 2:

0: 0C 94 34 00 jmp 0x68 ; 0x68 перебрасывает на строку 68 занимает 4 такта
4: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c

8: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
c: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
10: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
14: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
18: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
1c: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
20: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
24: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
28: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
2c: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
30: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
34: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
38: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
3c: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
40: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
44: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
48: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
4c: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
50: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
54: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
58: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
5c: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
60: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c
64: 0C 94 3E 00 jmp 0x7C ; 0x7c перебрасывает на строку 7c

68: 11 24 eor r1 r1 выполнение логического исключающего OR между содержимым регистра 1 и регистром 1 (обнулениие регистра 1) занимает 2 такта

6a: 1F BE out 0x3F , r1 ; 63 записывает значение регистра 1 в порт SREG

6c: CF EF ldi r28, 0xFF ; 255 записывает в регистр 28 значение 255

6e: D8 E0 ldi r29, 0x8 ; 8 записывает в регистр 29 значение 8

70: DE BF out 0x3E, r29 ; 62 заносит данные из 29 регистра в SPH

72: CD BF out 0x3D, r28 ; 61 заносит данные из 28 регистра в SPL

74: 0E 94 40 00 call 0x80 ; 0x80 команда call вызывает подпрограмму по адресу 0x80 и значения в SPH и SPL уменьшаются на 2

78: 0C 94 50 00 jmp 0xA0 ; 0xA0 перебрасывает на строку A0

7c: 0C 94 00 00 jmp 0 ; 0x00 перебрасывает на 0 строку

80: 38 9A sbi 0x7, 0 ; 7 устанавливает 0 бит на 7 регистр I/O

82: 3D 98 cbi 0x7, 5 ; 7 в 7 регистре очищается 5 бит

84: 45 9A sbi 0x8, 5 ; 8 устанавливает 5 бит на 8 регистр I/O

86: 35 99 sbic 0x6, 5 ; 6 проверяет состояние 5 бита в 6 регистре I/O и, если этот бит очищен, пропускает следующую команду

88: 02 C0 rjmp +4 ; 0x90 перебрасывает на 90 строку

8a: 40 9A sbi 0x8, 0; 8 устанавливает 0 бит на 8 регистр I/O

8c: 01 C0 rjmp +2 ; 0x90 перебрасывает на 90 строку

8e: 40 98 cbi 0x8, 0 ; 8 в 8 регистре очищается 0 бит

90: 28 E8 ldi r18, 0x88 ; 136 записывает в 18 регистр значение 136

92: 80 EF ldi r24, 0xF0 ; 240 записывает в 24 регистр значение 240

94: 91 E2 ldi r25, 0x21 ; 33 записывает в 25 регистр значение 33

96: 21 50 subi r18, 0x01 ; 1 вычитаем из 18 регистра 1
98: 80 40 sbci r24, 0x0 ; 0 вычитаем из 24 регистра флаг C
9a: 90 40 sbci r25, 0x0 ; 0 вычитаем из 25 регистра флаг C
9c: E1 F7 brne -8 ; 0x96 Условный относительный переход. Тестируется бит флага нулевого значения (Z) регистра статуса и, если бит очищен, выполняется переход относительно состояния счетчика программ.
9e: F3 CF rjmp -26 ; 0x86 относительный переход
a0: F8 94 cli очищает флаг глобального прерывания в регистре статуса
a2: FF CF rjmp -2 относительный переход

Подсчёт задержки:

Команды с 0x90 по 0x94 –загрузка константных значений в регистры

Команда 9c: e1 f7 brne .-8 является условием и в случае, если Z == 0, то есть результат предыдущего значения не равен нулю, переходим по указанному адресу 0x96.

Из этого следует, что команды с 0x96 по 0x9c являются циклом.

Первый вариант вычислений:
На первой итерации, где r18 = 136:
r18 = r18 - 1; 136 – 1 = 135 (C = 0)
r24 = r24 – 0 – C ; 240 – 0 – 0 = 240 (C = 0)
r25 = r25 – 0 – C; 33 – 0 – 0 = 33 (C = 0)
brne .-8 так как C == 0, переход в начало цикла

На 255 й итерации, где r18 = 0:
r18 = r18 - 1; 0 – 1 = 255 (C = 1)
r24 = r24– 0 – C; 240 – 0 – 1 = 239 (C = 0)
r25 = r25 – 0 – C; 33 – 0 – 0 = 33 (C = 0)
brne .-8 так как C == 0, переход в начало цикла

На N-й итерации, где r18 = 0, r24 = 0, r25 = 0:
r18 = r18 - 1; 0 – 1 = 255 (C = 1)
r24 = r24 – 0 – C; 0 – 0 – 1 = 255 (C = 1)
r25 = r25 – 0 – C; 0 – 0 – 1 = 255 (C = 1)
brne .-8 так как C == 1, переход на следующую команду

Так как subi – 1 цикл, sbci – 1 цикл, brne – 2 цикла пока в r25 >= 0, и 1 цикл на последней итерации, на всех итерациях, кроме одной требуется 5 циклов.

Общее количество циклов составит 5 * (256 * (256 * 33 + 240) + 136) – 1 = 11121319

rjmp .+0 добавит 2 цикла

nop добавит 1 цикл

Загрузка в регистры ещё 3 такта

Итого: 11121325 циклов

Расчитаем задержку: 11121325/16000000=0.695 сек или 695мс

Код на С:

```
#include <avr/io.h>
#include <util/delay.h>
#define F_CPU 16000000UL

Int main(void)
{
   DDRC |= (1 << 0);
   PORTC |= (1 << 5);
   while (1)
   {
      if ((PINC &(1 << 5)) == 0) PORTC |= (1 << PINC0);
      else PORTC &= ~(1 << PINC0);
      _delay_ms(695);
   }
}
```

Выводы

Мы научились восстанавливать ассемблерный код из HEX-кода используя только теоретические знания, а также составили алгоритм и написали код на С.