Simulasi Port I/O pada Atmega8535 Terhubung dengan Proteus dan Interrupt

Diperuntukkan untuk memenuhi salah satu tugas praktikum Mata Kuliah Aplikasi Mikrokontroler

Praktikum	: Aplikasi Mikrokontroler
Praktikum ke	: 7
Tanggal Praktikum	: Kamis, 12 November 2020
Tanggal Pengumpulan Laporan	: Rabu, 18 November 2020
Nama dan NIM	: 1. Amir Husein (181344003)
Kelas	: 3-TNK
Instruktur	: 1. Ferry Satria, BSEE., M.T
	2. Rahmawati Hasanah, S.ST., M.T

Politeknik Negeri Bandung

Tahun Ajaran 2020/2021

I. TUJUAN

- Mahasiswa dapat memahami prinsip penggunaan port pada Atmega8535
- Mahasiswa mampu melakukan koneksi antara simulator AVR Studio dengan simulator Proteus
- Mahasiswa dapat memahami dan mengaplikasikan penggunaan interrupt pada Atmega8535

II. LANDASAN TEORI

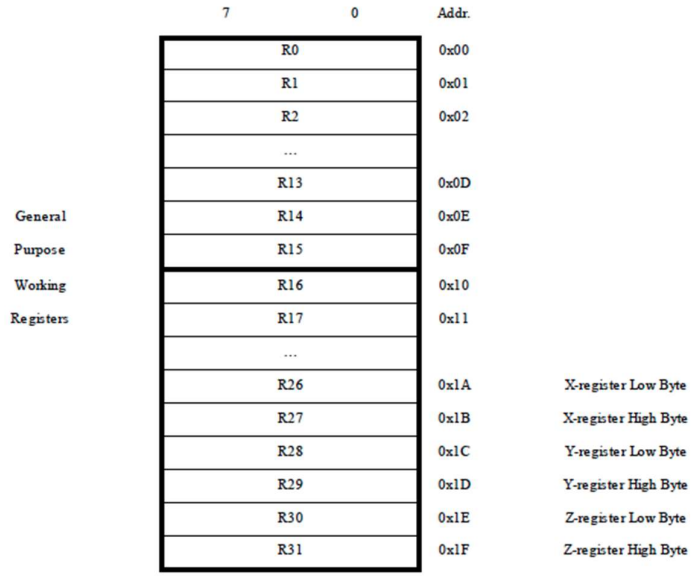
1. Mikrokontroler Atmega8535

   Mikrokontroler merupakan suatu chip komputer mini, dimana di dalamnya sudah terdapat sebuah mikprosesor disertai memori, baik itu RAM, ROM, maupun EEPROM. Selain itu, mikrokontroler telah memiliki sistem integrasi Input dan Output (I/O) yang telah dikemas sedemikian rupa pada rangkaian Integrated Circuitnya, sehingga memudahkan dalam melakukan tugas atau operasi tertentu.

   Atmega8535 merupakan sebuah mikrokontroler 8-bit yang dikeluarkan oleh perusahaan Atmel pada tahun 2006. Mikrokontroler ini pun memiliki flash memori sebesar 8kb serta EEPROM 512 byte. Selain itu, pada papan mikro ini pun sudah terdapat sebuah ADC dan 3 buah Timer sebagai pencacah waktu pemrosesan data.

2. General Purpose Register

   Register ini merupakan register umum yang dapat digunakan sebagai kontrol pengisian dan penyesuaian data. Pada Atmega8535, terdapat 32 general purpose register, yang penamaannya berupa rentang angka dari R0 sampai dengan R31.



Gambar 1. General Purpose Register pada Atmega8535

Terlihat pada gambar diatas, diantara R15 dan R16 terdpat sebuah garis tebal, hal ini menandakan bahwa register R15 kebawah tidak berlaku instruksi *immediate* atau

langsung, sedangkan register R16 keatas berlaku. Pada general purpose register ini juga terdapat register khusus sebagai pointer alamat 16-bit yaitu R2 hingga R21 yang terbagi menjadi 3 segmen yaitu pointer X, Y, dan Z.

3. Instruksi LPM dan LD

   Instruksi LPM (Load Program Memory) digunakan untuk melakukan *load* dari memori dengan tujuannya adalah register pointer. LPM digunakan untuk mendapatkan nilai yang tepat dari setiap alamat program memori yang dideklarasikan.

   Instruksi LD (Load from Data) digunakan untuk melakukan *load* dari memori, apabila LPM dari program area, maka LD dari data area.

4. I/O pada Atmega8535

   ATMEGA 8535 memilki empat buah port yang benar-benar memiliki fungsi untuk Read-Modify-Write ketika digunakan sebagai port General digital I/O. Driver pin pada I/O port cukup kuat untuk menggerakkan tampilan LED secara langsung. Semua pin port memiliki resistor pull-up yang dapat dipilih secara individual dengan resistansi invarian tegangan suplai.

   Dalam mengatur sebuah port I/O pada Atmega8535, kita perlu mengatur bit pada DDRX, X disini berdiri sebagai jenis port. Misal jika kita ingin mengatur port A, maka digunakan DDRA, begitupun dengan B hingga port D. Jika DDRX diberikan nilai 1 (high), maka pin digunakan sebagai output. Jika DDRX diberikan nilai 0 (low), maka pin difungsikan sebagai input. Selain DDRX, ada juga register PORTX digunakan untuk mengaktifkan pull-up resistor (pada saat pin difungsikan sebagai input), dan memberikan nilai keluaran pin high/low (pada saat difungsikan sebagai output). Terdapat juga PINX, sebuah register yang berfungsi untuk mengetahui keadaan tiap-tiap pin pada mikrokontroller.

5. Interrupt pada Atmega8535

   Pada Atmega8535 terdapat 3 buah eksternal interrupt, yaitu sebagai berikut dengan disertai alamat vectornya:

   1. INT0 === 001
   2. INT1 === 002
   3. INT2 === 012

   Untuk dapat menggunakan interrupt ini, langkah yang perlu dilakukan pertama kali ialah memastikan bahwa register ‘I’ (Interrupt) pada SREG bernilai 1, agar interrupt enable, sebaliknya bila diisikan 0, maka interrupt akan disable.

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

   Selain itu, terdapat GICR (General Interrupt Control Register) yang berfungsi untuk mengontrol penempatan dari interrupt vector, dan untuk milih interrupt mana yang akan diaktifkan. Untuk meng-aktifkan INT1 maka bit 7 harus diisi 1, untuk meng-aktifkan

INT0 maka bit 6 harus diisi 1, dan untuk meng-aktifkan INT2 maka bit 5 harus diisi 1. Sebaliknya, apabila diisi dengan logika 0, maka interrupt yang bersangkutan akan disable. Untuk struktur GICR dapat dilihat pada gambar dibawah ini:

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Terdapat juga MCUCR (MicroController Unit Control Register)yang berisi bit kontrol untuk mode interrupt INT0 dan INT1. MCUCR ini berguna untuk menentukan tipikal mode bagi interrupt INT0 dan INT1, seperti bagaimana respon dalma menerima input untuk dapat diproses pada mikrokontroler. Untuk struktur MCUCR dapat dilihat pada gambar dibawah ini:

Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 0 dan Bit 1 pada MCUCR berkaitan dengan INT0 dan Bit 2 serta 4 berkaitan dengan INT1, mode dari interrupt dapat diatur sedemikian rupa sesuai pada tabel kombinasi ISC. Untuk INT0 ditunjukkan sebagai berikut:

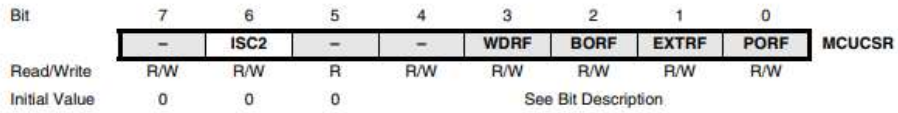
ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

Dan tabel dibawah ini untuk INT1:

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

Sebagai contoh, untuk INT1, apabila ISC11 diatur pada logika 0 dan ISC10 pada logika 1, maka INT1 akan bekerja ketika menerima setiap perubahan logika, sedangkan bila ISC11 dan ISC10 diatur 1, maka INT1 akan bekerja ketika rising edge atau ada peningkatan nilai logika.

Kemudian untuk mengatur mode pada INT2, digunakan MCUSCR (MicroController Unit Control and Status Register) pada bit ke-6, seperti berikut :



Bit	7	6	5	4	3	2	1	0	
	–	ISC2	–	–	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

III. SOAL LAB

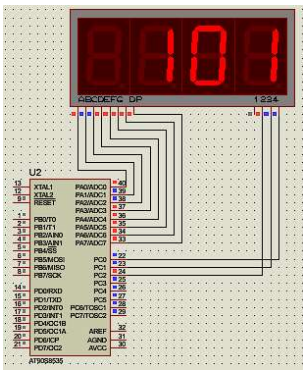
1. Buat program untuk Heksa-Desimal up-counter 3 digit (menghitung dari $100 sampai $30F) yang outputnya berubah 1/5 detik sekali. Tidak berulang.
2. Buat program untuk Heksa-Desimal up-counter 3 digit (menghitung dari $100 sampai $30F) yang outputnya berubah 1/5 detik sekali. Berulang tanpa henti.
3. Buat program untuk Desimal up-counter 3 digit (menghitung dari 100 sampai 359) yang outputnya berubah 1/5 detik sekali. Tidak berulang.
4. Buat program untuk Desimal up-counter 3 digit (menghitung dari 100 sampai 359) yang outputnya berubah 1/5 detik sekali. Berulang tanpa henti.
5. Buat program untuk Octal up-counter 3 digit (menghitung dari 100 sampai 359) yang outputnya berubah 1/5 detik sekali. Tidak berulang.
6. Buat program untuk octal up-counter 3 digit (menghitung dari 100 sampai 359) yang outputnya berubah 1/5 detik sekali. Berulang tanpa henti.
7. Buat program untuk Binary up-counter 5 bit yang outputnya berubah 1 detik sekali, menghitung berulang terusmenerus. Gunakan unit peraga 7 segment sebagai output.
8. Buat program untuk Binary up-counter 8 bit yang outputnya berubah 1/4 detik sekali, menghitung berulang terusmenerus. Gunakan unit peraga 7 segment sebagai output.
9. Buat program untuk menampilkan 2 kata pada unit peraga 7 segmen (8 karakter) secara bergantian berulang, masing-masing selama 4 detik. Kata pertama: “rAJInLAH”. Kata ke dua: “bELAJAr-”.
10. Buat program untuk menampilkan kata: “--rAJIn-”,”-JUJUr--”, dan “—tULUS-” pada unit peraga 7 segmen (8 karakter) secara bergantian berulang, masing-masing selama 4 detik.
11. Buat program untuk menampilkan kata: “--rAJIn-”,”-JUJUr--”, dan “—tULUS-” pada unit peraga 7 segmen (8 karakter) secara bergantian berulang. Pergantian kata dikendalikan melalui tombol interupsi int0.
12. Buat program untuk menampilkan kata: “--rAJIn-”,”-JUJUr--”, dan “—tULUS-” pada unit peraga 7 segmen (8 karakter) secara bergantian berulang, masing-masing selama 4 detik. Kata “-JUJUr--” berkedip 2 kali tiap detik.

IV. PROGRAM & HASIL

1. Program:

```
.INCLUDE "M8535DEF.INC"
.ORG 0
//PEMROGRAM      : AMIR HUSEIN
//NIM            : 181344003
RJMP MAIN
MAIN:   LDI R16, LOW(RAMEND)
        OUT SPL, R16
        LDI R16, HIGH(RAMEND)
        OUT SPH, R16
        LDI R20, 0X0FF
        OUT DDRC, R20
        OUT DDRA, R20
        LDI ZL, LOW(2*DAT)
        LDI ZH, HIGH(2*DAT)
        LDI R17, 01
        LDI R16, $00
S33:    LDI R24, 12
S11:    LDI R19, 0X04
        OUT PORTC, R19
        MOV ZL, R17
        LPM R18, Z
        OUT PORTA, R18
        RCALL DELAY
        LDI R19, 0X02
        OUT PORTC, R19
        MOV R19, R16
        ANDI R19, 0XF0
        SWAP R19
        MOV ZL, R19
        LPM R18, Z
        OUT PORTA, R18
        RCALL DELAY
        LDI R19, 0X01
        OUT PORTC, R19
        MOV R19, R16
        ANDI R19, 0X0F
        MOV ZL, R19
        LPM R18, Z
        OUT PORTA, R18
        RCALL DELAY
        DEC R24
        BRNE S11
        LDI R27, 1
        ADD R16, R27
        ADC R17, R5
        CPI R17, 03
        BRNE S33
        CPI R16, 0X10
        BRNE S33

AKHIR:  RJMP AKHIR

DELAY:  PUSH R21
        PUSH R22
        PUSH R23
        LDI R21, 1
LUP21:  LDI R22, 0X22
LUP22:  LDI R23, 0
LUP23:  DEC R23
        BRNE LUP23
        DEC R22
        BRNE LUP22
        DEC R21
        BRNE LUP21
        POP R23
        POP R22
        POP R21
        RET

.ORG 80
DAT:
.DB 0X3F,0X06,0X5B,0X4F,0X66,0X6D,0X7D,0X07,0X7F,0X6F,0X77,0X7C,0X39,0X5E,0X79,0X71
```

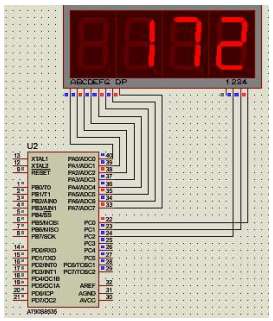
Hasil:



2. Program:

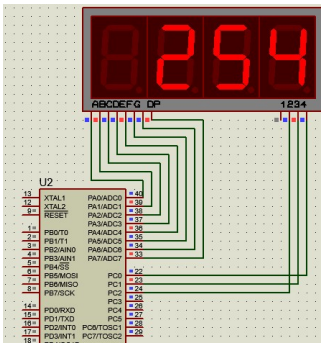
```
//no 2
//upcounter HD 3 digit dengan Multiple Segment
//berulang
.include "m8535def.inc"
.org 0x0000
rjmp main
main:

ldi r16,low(ramend)
out spl,r16
ldi r16,high(ramend)
out sph,r16

ulang: ldi r20,0x0ff
out ddra,r20
out ddrc,r20
ldi zl,low(2*DAT)
ldi zh,high(2*DAT)

ldi r17,$1  //bit 1
ldi r16,$00
s22:ldi r24,12
s11: ldi r19,$04
     ldi r19,4  //mengaktifkan posisi 7 seg dari ketiga dari kanan
     out portc,r19


mov zl,r17
lpm r18,z+          //tampilkan counter
out porta,r18       //high byte
rcall d1s
ldi r19,0x02
out portc,r19
mov r19,r16
andi r19,0x0f0
swap r19
mov zl,r19
lpm r18,z+
out porta,r18
rcall d1s

ldi r19,$1
out portc,r19
mov r19,r16
andi r19,$0f
mov zl,r19
lpm r18,z+
out porta,r18
rcall d1s
dec r24
brne s11
ldi r27,$1
add r16,r27     //menaikan counter
adc r17,r5
cpi r17,3
brne s22
cpi r16,$10
brne s22

rjmp ulang
//ldi r27,$0FF
//out porta,r27

end:rjmp end

;================================
d1s:PUSH R21 ;SUBRUTIN DELAY 1 DETIK
     PUSH R22
     PUSH R23
     LDI R21,1
L11:LDI R22,$22
L22:LDI R23,0
L33:DEC R23
    BRNE L33
    DEC R22
    BRNE L22
    DEC R21
    BRNE L11
    POP R23
    POP R22
    POP R21
    RET


.org 0x80
//pola karakter
DAT:
    .db $C0,$F9,$A4,$B0,$99,$92,$82,$F8,$80,$90,$88,$83,$C6,$A1,$86,$8E
```

Hasil:



3. Program:

Hasil:

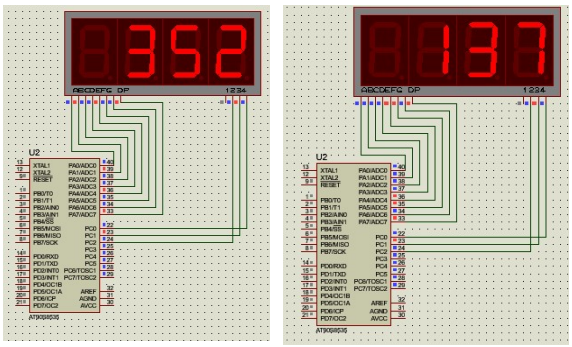



```
;NO.3
;UP COUNTER 3 DGT DES TIDA BERULANG

.INCLUDE"M8535DEF.INC"
.ORG 0
RJMP MAIN

MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

LDI R20,0X0FF
OUT DDRA,R20
LDI R20,0X0FF
OUT DDRC,R20

LDI ZL,LOW(KAR*2)
LDI ZH,HIGH(KAR*2)

LDI R17,01      ;DIGIT 3 MULAI DR 1
LDI R16,00      ;BUAT POLKAR DGT 2 DAN 1

B: LDI R19,0X04     ;POSISI DIGIT KETIGA
   OUT PORTC, R19
   MOV ZL,R17       ;POL KAR
   LPM R18,Z
   OUT PORTA,R18
   RCALL DELAY5MS

LDI R19,0X02    ;POSISI DIGIT 2
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F0
SWAP R19
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R19,0X01    ;POSISI DIGIT 1
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R27,1
ADD R16,R27
MOV R19,R16
RCALL DAA
MOV R16,R19
ADC R17,R5
CPI R17,3
BRNE B
CPI R16,$60
BRNE B
BREQ AKHIR

AKHIR: RJMP AKHIR
DELAY5MS: LDI R21, 1    ;DELAY
L21:   LDI R22, 0
L22:   LDI R23, 0
L23:   DEC R23
       BRNE L23
       DEC R22
       BRNE L22
       DEC R21
       BRNE L21
       RET
;-------------------------------------
daa: push r16;amankan isi reg. ke stack

push r17
push r21
push r24
in r16,sreg; amankan sreg
in r17,sreg
ldi r24,0 ; r24 hanya sbg status dr carry flag
adc r24,r24; status carry flag sebelum koreksi

; ambil, amankan kembali sreg, uji half carry

out sreg,r17
brhc lup4; lompat ke lup4 jika H=0
rcall fk_ln; satuan + 06 jika H=1
; rjmp lup5
lup4: mov r21,r19

andi r21,$0f; ambil dgt.satuan
cpi r21,$0a ; > 9
brcs lup5; lompat ke lup5 jika < 9
rcall fk_ln; satuan + 06 jika > 9

lup5: out sreg,r17; ambil, amankan kembali sreg, uji carry

brcc lup6; lompat ke lup6 jika C=0
rcall fk_hn; puluhan + 60 jika C=1
; rjmp lup7
lup6: mov r21,r19

andi r21,0xf0; ambil dgt.puluhan
cpi r21,0xa0
brcs lup7; lompat ke lup7 jika pul < a
rcall fk_hn; pul + 60 jika > 9

lup7: out sreg,r16; mengatur carry flag sblm keluar:

clc; C = 0
in r16,sreg ;amankan sreg stlh c=0
cpi r24,0 ;menguji status
breq lup8
out sreg,r16;ambil sreg, set c
sec; C = 1
in r16,sreg; amankan sreg stlh c=1
lup8: out sreg,r16;status akhir
pop r24;kembalikan isi reg. dr stack
pop r21
pop r17
pop r16
ret

fk_ln: ldi r21,06; koreksi dgt.satuan
out sreg,r16
add r19,r21
in r16,sreg
brcc tuh
ldi r21,0x01
add r17,r21
tuh: out sreg,r16
adc r24,r24
ret

fk_hn: ldi r21,$60; koreksi dgt.puluhan

out sreg,r16
add r19,r21
in r16,sreg
adc r24,r24; amankan carry pd r24
ret ; akhir dari prog. subrutin.
;-----------------------------------------------

.ORG $80
KAR:
.DB $C0,$F9,$A4,$B0,$99,$92,$82,$F8,$80,$90
```

4. Program:

```
;NO.4
;UP COUNTER 3 DGT DES BERULANG

.INCLUDE"M8535DEF.INC"
.ORG 0
RJMP MAIN

MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

LDI R20,0X0FF
OUT DDRA,R20
LDI R20,0X0FF
OUT DDRC,R20

LDI ZL,LOW(KAR*2)
LDI ZH,HIGH(KAR*2)

A: LDI R17,01       ;DIGIT 3 MULAI DR 1
   LDI R16,00       ;BUAT POLKAR DGT 2 DAN 1

B: LDI R19,0X04     ;POSISI DIGIT KETIGA
   OUT PORTC, R19
   MOV ZL,R17       ;POL KAR
   LPM R18,Z
   OUT PORTA,R18
   RCALL DELAY5MS

LDI R19,0X02    ;POSISI DIGIT 2
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F0
SWAP R19
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R19,0X01    ;POSISI DIGIT 1
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R27,1
ADD R16,R27
MOV R19,R16
RCALL DAA
MOV R16,R19
ADC R17,R5
CPI R17,3
BRNE B
CPI R16,$60
BRNE B
RJMP A

AKHIR: RJMP AKHIR
```

Hasil:

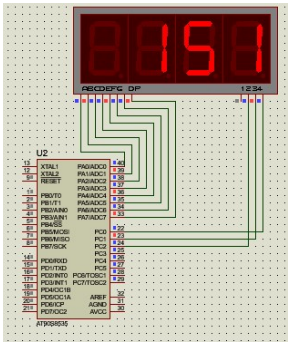


5. Program:

```
;NO.5
;UP COUNTER 3 DGT OKT TIDA BERULANG

.INCLUDE"M8535DEF.INC"
.ORG 0
RJMP MAIN

MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

LDI R20,0X0FF
OUT DDRA,R20
LDI R20,0X0FF
OUT DDRC,R20

LDI ZL,LOW(KAR*2)
LDI ZH,HIGH(KAR*2)

LDI R17,01       ;DIGIT 3 MULAI DR 1
LDI R16,00       ;BUAT POLKAR DGT 2 DAN 1

B: LDI R19,0X04     ;POSISI DIGIT KETIGA
   OUT PORTC, R19
   MOV ZL,R17       ;POL KAR
   LPM R18,Z
   OUT PORTA,R18
   RCALL DELAY5MS

LDI R19,0X02    ;POSISI DIGIT 2
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F0
SWAP R19
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R19,0X01    ;POSISI DIGIT 1
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R27,1
ADD R16,R27
RCALL KOR
CPI R17,3
BRNE B
CPI R16,$60
BRNE B
BREQ AKHIR

AKHIR: RJMP AKHIR
```

```
DELAY5MS: LDI R21, 1    ;DELAY
L21:    LDI R22, 0
L22:    LDI R23, 0
L23:    DEC R23
        BRNE L23
        DEC R22
        BRNE L22
        DEC R21
        BRNE L21
        RET

KOR:
MOV R29,R16
ANDI R29,$0F
CPI R29,8
BRLO KOR2
LDI R30,8
ADD R16,R30
KOR2:
MOV R29,R16
ANDI R29,$0F0
CPI R29,$80
BRLO DAH
LDI R30,$80
ADD R16,R30
ADC R17,R5
DAH:
RET


.ORG $80
KAR:
.DB $C0,$F9,$A4,$B0,$99,$92,$82,$F8
```

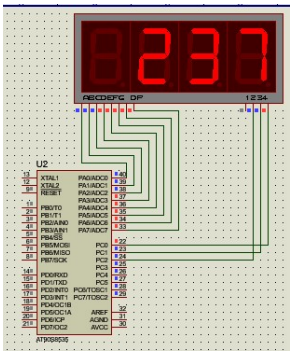
Hasil:



6. Program:

```
;NO.6
;UP COUNTER 3 DGT OKT TIDA BERULANG

.INCLUDE"M8535DEF.INC"
.ORG 0
RJMP MAIN

MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

LDI R20,0X0FF
OUT DDRA,R20
LDI R20,0X0FF
OUT DDRC,R20

LDI ZL,LOW(KAR*2)
LDI ZH,HIGH(KAR*2)

A: LDI R17,01       ;DIGIT 3 MULAI DR 1
   LDI R16,00       ;BUAT POLKAR DGT 2 DAN 1

B: LDI R19,0X04     ;POSISI DIGIT KETIGA
   OUT PORTC, R19
   MOV ZL,R17       ;POL KAR
   LPM R18,Z
   OUT PORTA,R18
   RCALL DELAY5MS

LDI R19,0X02    ;POSISI DIGIT 2
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F0
SWAP R19
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R19,0X01    ;POSISI DIGIT 1
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R27,1
ADD R16,R27
RCALL KOR
CPI R17,3
BRNE B
CPI R16,$60
BRNE B
BRNE B
RJMP A

AKHIR: RJMP AKHIR

DELAY5MS: LDI R21, 1    ;DELAY
L21:    LDI R22, 0
L22:    LDI R23, 0
L23:    DEC R23
        BRNE L23
        DEC R22
        BRNE L22
        DEC R21
        BRNE L21
        RET

KOR:
MOV R29,R16
ANDI R29,$0F
CPI R29,8
BRLO KOR2
LDI R30,8
ADD R16,R30
KOR2:
MOV R29,R16
ANDI R29,$0F0
CPI R29,$80
BRLO DAH
LDI R30,$80
ADD R16,R30
ADC R17,R5
DAH:
RET

.ORG $80
KAR:
.DB $C0,$F9,$A4,$B0,$99,$92,$82,$F8
```

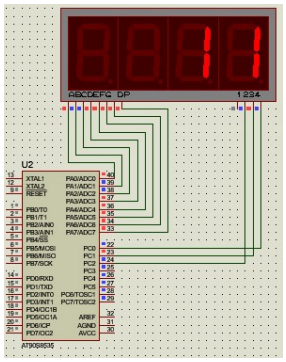
Hasil:



7. Program:

```
;NO.7
;BINARY UP COUNT 5 BIT
;OUTPUT 7SEG

.INCLUDE"M8535DEF.INC"
.ORG 0
RJMP MAIN

MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

LDI R20,0X0FF
OUT DDRA,R20
LDI R20,0X0FF
OUT DDRC,R20

LDI ZL,LOW(KAR*2)
LDI ZH,HIGH(KAR*2)

A:
LDI R16,00      ;BUAT POLKAR DGT 2 DAN 1

B:
LDI R19,0X02    ;POSISI DIGIT KIRI
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F0
SWAP R19
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R19,0X01    ;POSISI DIGIT KANAN
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R27,1
ADD R16,R27
CPI R16,$20
BRNE B
RJMP A

AKHIR: RJMP AKHIR
DELAY5MS: LDI R21, 1    ;DELAY
L21:    LDI R22, 0
L22:    LDI R23, 0
L23:    DEC R23
        BRNE L23
        DEC R22
        BRNE L22
        DEC R21
        BRNE L21
        RET
.ORG $80
KAR:
.DB $C0,$F9,$A4,$B0,$99,$92,$82,$F8,$80,$90,$88,$83,$C6,$A1,$86,$8E
```

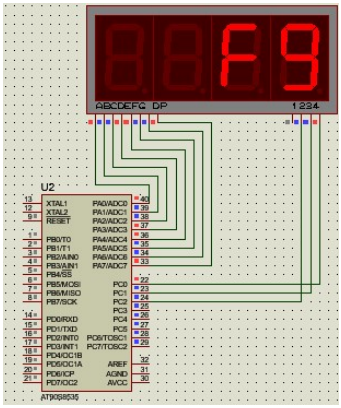
Hasil:



8. Program:

```
;NO.8
;BINARY UP COUNT 8 BIT
;OUTPUT 7SEG

.INCLUDE"M8535DEF.INC"
.ORG 0
RJMP MAIN

MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

LDI R20,0X0FF
OUT DDRA,R20
LDI R20,0X0FF
OUT DDRC,R20

LDI ZL,LOW(KAR*2)
LDI ZH,HIGH(KAR*2)

A:
LDI R16,00      ;BUAT POLKAR DGT 2 DAN 1

B:
LDI R19,0X02    ;POSISI DIGIT KIRI
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F0
SWAP R19
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R19,0X01    ;POSISI DIGIT KANAN
OUT PORTC,R19
MOV R19,R16
ANDI R19,0X0F
MOV ZL,R19
LPM R18,Z
OUT PORTA,R18
RCALL DELAY5MS

LDI R27,1
ADD R16,R27
CPI R16,$00
BRNE B
RJMP A

AKHIR: RJMP AKHIR
```

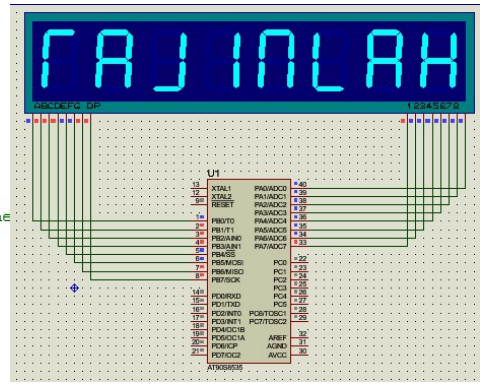
Hasil:

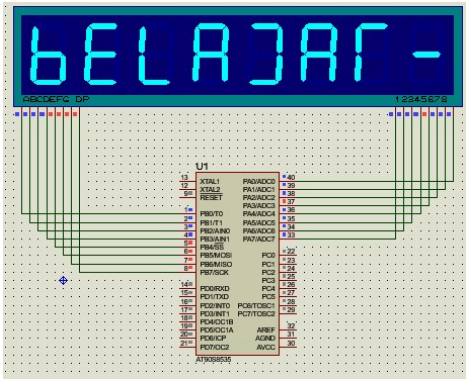



```
AKHIR: RJMP AKHIR
DELAY5MS: LDI R21, 1    ;DELAY
L21:    LDI R22, 0
L22:    LDI R23, 0
L23:    DEC R23
        BRNE L23
        DEC R22
        BRNE L22
        DEC R21
        BRNE L21
        RET
.ORG $80
KAR:
.DB $C0,$F9,$A4,$B0,$99,$92,$82,$F8,$80,$90,$88,$83,$C6,$A1,$86,$8E
```

9. Program:

Hasil:



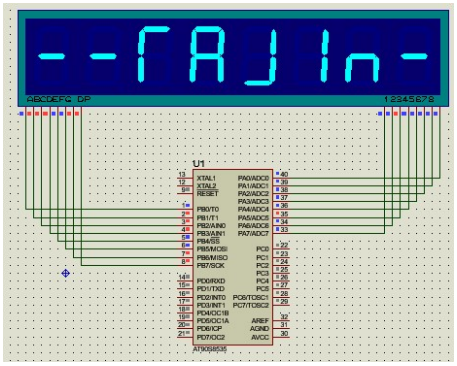


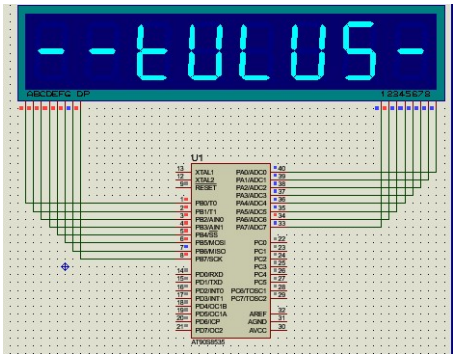

```
//no9
.include "m8535def.inc"
.org 0x0000
rjmp main
main:

ldi r16,low(ramend)
out spl,r16
ldi r16,high(ramend)
out sph,r16

ldi r20,0x0ff
out ddra,r20
out ddrb,r20

here: ldi r16,250

lup: ldi zl,low(2*DAT)
     ldi zh,high(2*DAT)
     ldi r23,0x8    //counter, terdapat 8 kar dalam 1 frame

tampil: lpm r21,z+
        out porta,r21 //counter posisi digit
        lpm r22,z+
        out portb,r22
        rcall delay
        dec r23
        brne tampil
        dec r16
        brne lup



        ldi r17,0x0ff
        out portb, r17
        rcall dls

        ldi r16,240
        ldi zl,0
        ldi zh,0

lup2:   ldi zl,low(2*DAT2)
        ldi zh,high(2*DAT2)
        ldi r23,0x8 //counter, terdapat 8 kar dalam 1 frame

tampil2:lpm r21,z+
        out porta,r21 //counter posisi digit
        lpm r22,z+
        out portb,r22
        rcall delay
        dec r23
        brne tampil2
        dec r16
        brne lup2
        ldi r17,0x0ff
        out portb, r17
        rcall dls


        rjmp here

end: rjmp end
delay:  ldi r25,0x2
    L1: ldi r24,0xaa
    L2: dec r24
        brne L2
        dec r25
        brne L1
        ret

dls: ldi r21,4  //subrutin delay 1s
    K21:ldi r22,200
    K22:ldi r23,200

    K23:nop
        nop
        nop
        nop

        dec r23     //mengurang r23
        brne K23
        nop
        dec r22     //mengurang r22
        brne K22
        nop
        dec r21     //mengurang r21
        brne K21
        nop
        ret
.org 0x80
DAT:
    .db $80,$83,$40,$86,$20,$C7,$10,$88,$8,$F0,$4,$88,$2,$ce,$1,$BF
DAT2:
    .db $80,$CE,$40,$88,$20,$F1,$10,$f9,$8,$c8,$4,$C7,$2,$88,$1,$89
```

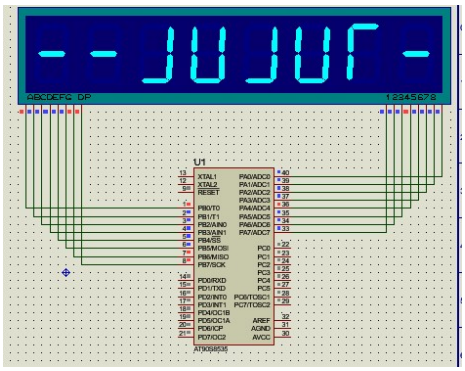
10. Program:

```
//template proteus
;menampilkan RAJINLAH, Belajar, dan tulus pada 7 segment
;dan Terlihat bersamaan selama 4 detik GANTIAN

.INCLUDE"M8535DEF.INC"
.ORG 0
RJMP MAIN

MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

LDI R20,0XFF
OUT DDRB,R20
OUT DDRA,R20

YU:
LDI R16,240
TAMPIL:
LDI ZL,LOW(2*DAT)
LDI ZH,HIGH(2*DAT)
LDI R23,0X8          ;COUNTER 1 FRAME 8 KAR

RAJIN:
LPM R21,Z+
OUT PORTA,R21        ;NGATU RPERGANTIAN POSISI DIGIT
LPM R22,Z+
OUT PORTB,R22        ;NGATUR KARAKTER
RCALL TUNDA          ;DELAY 2MS
DEC R23              ;DEC COUNTER
BRNE RAJIN
DEC R16
BRNE TAMPIL          ;ULANG SAMPE BODO
;CLEAR DISPLAY
LDI R17,$FF
OUT PORTB,R17
RCALL TUNDA

LDI R16,240
ITUNG:
LDI ZL,LOW(2*DAT1)
LDI ZH,HIGH(2*DAT1)
LDI R23,0X8          ;COUNTER 1 FRAME 8 KAR

BLJR:
LPM R21,Z+
OUT PORTA,R21        ;NGATU RPERGANTIAN POSISI DIGIT
LPM R22,Z+
OUT PORTB,R22        ;NGATUR KARAKTER
RCALL TUNDA          ;DELAY 2MS
DEC R23              ;DEC COUNTER
BRNE BLJR
DEC R16
BRNE ITUNG
;CLEAR DISPLAY
LDI R17,$FF
OUT PORTB,R17
RCALL TUNDA
LDI R16,240
LAGI:
LDI ZL,LOW(2*DAT2)
LDI ZH,HIGH(2*DAT2)
LDI R23,0X8          ;COUNTER 1 FRAME 8 KAR

TULUS:
LPM R21,Z+
OUT PORTA,R21        ;NGATU RPERGANTIAN POSISI DIGIT
LPM R22,Z+
OUT PORTB,R22        ;NGATUR KARAKTER
RCALL TUNDA          ;DELAY 2MS
DEC R23              ;DEC COUNTER
BRNE TULUS
DEC R16
BRNE LAGI            ;ULANG SAMPE BODO
;CLEAR DISPLAY
LDI R17,$FF
OUT PORTB,R17
RCALL TUNDA
RJMP YU

AKHIR: RJMP AKHIR
TUNDA: LDI R25, 3    ;DELAY
L21:   LDI R24, 0XAA
L23:   DEC R24
       BRNE L23
       DEC R25
       BRNE L21
       RET

DAT:  .DB $80,$BF,$40,$BF,$20,$CE,$10,$88,$08,$F1,$04,$F9,$02,$AB,$01,$BF
DAT1:  .DB $80,$BF,$40,$BF,$20,$87,$10,$C1,$08,$C7,$04,$C1,$02,$92,$01,$BF
DAT2:  .DB $80,$BF,$40,$BF,$20,$F1,$10,$C1,$08,$F1,$04,$C1,$02,$CE,$01,$BF
```

Hasil:



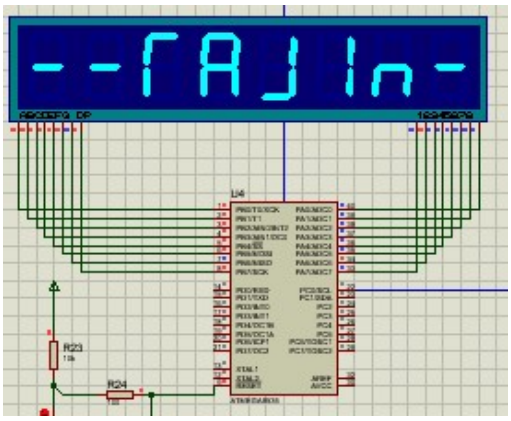


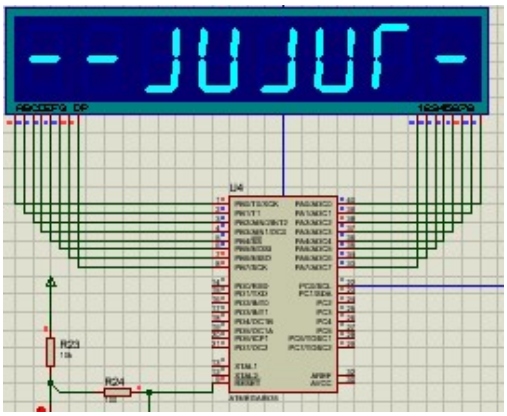


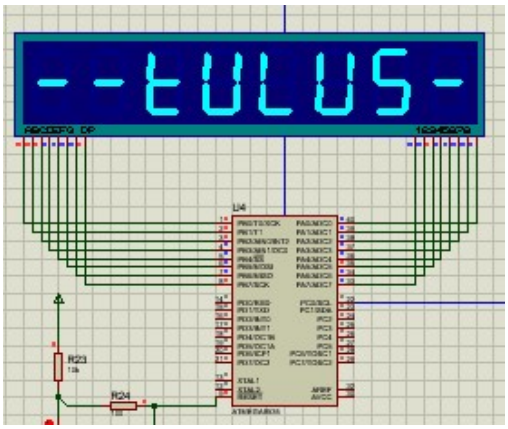
12. Program:

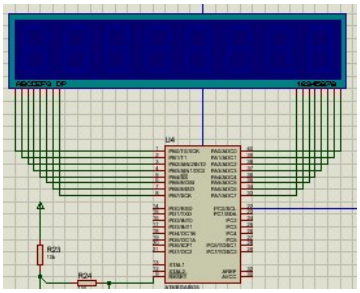
```
.INCLUDE"M8535DEF.INC"
.ORG 0
RJMP MAIN

MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

LDI R20,0XFF
OUT DDRB,R20
OUT DDRA,R20

YU:
LDI R16,240
TAMPIL:
LDI ZL,LOW(2*DAT)
LDI ZH,HIGH(2*DAT)
LDI R23,0X8          ;COUNTER 1 FRAME 8 KAR

RAJIN:
LPM R21,Z+
OUT PORTA,R21        ;NGATU RPERGANTIAN POSISI DIGIT
LPM R22,Z+
OUT PORTB,R22        ;NGATUR KARAKTER
RCALL TUNDA          ;DELAY 2MS
DEC R23              ;DEC COUNTER
BRNE RAJIN
DEC R16
BRNE TAMPIL          ;ULANG
;CLEAR DISPLAY
LDI R17,$FF
OUT PORTB,R17
RCALL TUNDA

LDI R18,$8
REP:
LDI R16,30
ITUNG:
LDI ZL,LOW(2*DAT1)
LDI ZH,HIGH(2*DAT1)
LDI R23,0X8          ;COUNTER 1 FRAME 8 KAR

JUJUR:
LPM R21,Z+
OUT PORTA,R21        ;NGATUR RPERGANTIAN POSISI DIGIT
LPM R22,Z+
OUT PORTB,R22        ;NGATUR KARAKTER
RCALL TUNDA          ;DELAY 2MS
DEC R23              ;DEC COUNTER
BRNE JUJUR
DEC R16
BRNE ITUNG
;CLEAR DISPLAY
LDI R17,$FF
OUT PORTB,R17
RCALL ONEDTK
DEC R18
BRNE REP
LDI R16,240
LAGI:
LDI ZL,LOW(2*DAT2)
LDI ZH,HIGH(2*DAT2)
LDI R23,0X8          ;COUNTER 1 FRAME 8 KAR

TULUS:
LPM R21,Z+
OUT PORTA,R21        ;NGATU RPERGANTIAN POSISI DIGIT
LPM R22,Z+
OUT PORTB,R22        ;NGATUR KARAKTER
RCALL TUNDA          ;DELAY 2MS
DEC R23              ;DEC COUNTER
BRNE TULUS
DEC R16
BRNE LAGI            ;ULANG SAMPE BODO
;CLEAR DISPLAY
LDI R17,$FF
OUT PORTB,R17
RCALL TUNDA
RJMP YU

AKHIR: RJMP AKHIR
TUNDA: LDI R25, 3    ;DELAY
L21:   LDI R24, 0XAA
L23:   DEC R24
       BRNE L23
       DEC R25
       BRNE L21
       RET

ONEDTK : LDI R25,1   ;DELAY
L1:    LDI R24,0
L2:    LDI R23,0
L3:    DEC R23
       BRNE L3
       DEC R24
       BRNE L2
       DEC R25
       BRNE L1
       RET

DAT: .DB $80,$BF,$40,$BF,$20,$CE,$10,$88,$08,$F1,$04,$F9,$02,$AB,$01,$BF
DAT1: .DB $80,$BF,$40,$BF,$20,$F1,$10,$C1,$08,$F1,$04,$C1,$02,$CE,$01,$BF
DAT2: .DB $80,$BF,$40,$BF,$20,$87,$10,$C1,$08,$C7,$04,$C1,$02,$92,$01,$BF
```

Hasil:









V. ANALISIS

1. Pada Program ini, akan dibuat sebuah counter 3 digit heksadesimal dari 100 – 30F. Digunakan multiple 7 segment berjenis common anode. Bit paling rendah disimpan paling kanan dengan kode posisi digit yaitu $1 dan bit paling besar disimpan di paling kiri dengan kode posisi digit $4. Pada Rangkaian, hubungkan port C ke pin 4,3,2 dan port A dihubungkan dengan pin A-G,Dp. Kedua port ini bertindak sebagai output maka itu diberi nilai FF agar dapat bertindak sebagai output. Digunakan program area DAT yang berisi kode posisi dan kode karakter heksadesimal (0-F).

   Selain itu, Port C akan bertindak sebagai output kode posisi digit dan Port A sebagai output kode karakter. Dari soal, digit paling kiri langsung di outputkan "1", digit kiri ini akan digunakan counter R17. Sementara itu, untuk digit tengah dan paling kanan digunakan register R16 sebagai counternya. Dimulai dengan memprogram digit paling kiri, maka dimasukan kode posisinya yaitu $4 dioutputkan ke port C. Untuk melakukan *load* data dari program area, digunakan register R18 untuk mengambil kode karakter "1" yang lalu di outputkan ke port A sehingga tercetak 1 pada 7 segment paling kiri.

   Selanjutnya, perlu dilakukan konfigurasi untuk memprogram digit tengah dengan kode posisinya $2. Digunakan counter R16 yang berinisiasi 00, lalu di load program memory nya untuk mengambil kode karakter "0" di outputkan ke port C sehingga muncul pada digit tengah. Counter R16 ini digunakan juga untuk digit paling kanan dengan status output yang berbeda. Maka dari itu nilai di R16 nya ini harus diambil perenambelasannya saja. Caranya dengan memindahkannya ke R19 yang lalu di beri logic AND dengan 0xF0 sehingga terambil perenambelasannya, tetapi nilai ini masih berada di posisi perenambelasan, maka dari itu digunakan SWAP sehingga tertukar dengan low nibble nya, sehingga posisi nya menjadi di satuan kembali. Data yang ada di satuan ini akan menjadi alamat pointer z, lalu di load datanya (kode karakter) di R18 lalu dioutputkan ke digit tengah. Dilanjutkan untuk memprogram digit 3 atau paling kanan, Digunakan kode posisi $1 selaku posisi digit paling kanan. Digunakan counter R16 sebelumnya yang lalu di proses pada R19 untuk diambil satuannya dengan AND $0F. Tidak perlu di swap karena sudah pada posisi satuan. Nilai tersebut dijadikan counter yang lalu dilakukan load data ke R18 yang di outputkan ke Port C sehingga tercetak di 7 segment paling kanan.

Dengan menaruh pola karakter pada alamat yang sama dengan karakter kita bisa dengan mudah menggunakan 1 register sebagai counter dan juga alamat sekaligus. Misalnya pertama kita ingin mengeluarkan 00, maka kita bisa isikan 00 ke suatu register kemudian satuan dijadikan alamat untuk mengambil data dan begitu juga dengan puluhan karena data tersebut akan ada pada alamat yang sama dengan datanya. Kemudian jika kita ingin mengeluarkan data 12 maka data tersebut bisa dijadikan alamat yang sudah mengandung pola karakter dari data tersbeut. Begitu lah yang dilakukan dengan R16 sebagai counter dan alamat sekaligus untuk digit satuan dan puluhan dengan cara yang sudah di jelaskan di paragraf sebelumnya. Agar nilai counter R16 dan R17 terus meningkat, nilai R16 ini ditambahkan 1 setiap selesai proses dan carry nya di tambahkan ke R17 selaku counter ratusan. Untuk menghentikan counter, dilakukan counter digit paling kiri dengan $3 atau R17 dengan $3. Apabila belum mencapai $3, maka digit tengah dan digit paling kanan terus melakukan counter dan prosesnya (diloncatkan kembali ke s22). Apabila sudah mencapai $3, maka pembanding selanjutnya adalah $10 untuk digit tengah dan akhirnya, yang dimana data terakhir nya menjadi 30F. Apabila persyaratan ini sudah tercapai, masukan nilai FF ke r27 dan dioutputkan ke port A sehingga semua 7 segment menjadi mati, tandanya program sudah selesai.

2. Program ini tidak jauh berbeda dari program nomor 1, hanya saja perlu ditambahkan pengulangan yaitu dengan cara menambahkan label “ulang” diawal program (mulai dari mengatur port A dan C sebagai output) lalu pada akhir program ditambahkan RJMP ULANG agar program terus berulang. Bagian pengisian data R27 dengan FF untuk mematikan 7 segment dihilangkan agar counter terus berulang tidak berhenti.

3. Pada program ini, sebenarnya tidak jauh berbeda dengan program pada nomor sebelumnya yaitu nomor 1 dan 2. Letak perbedaannya ada pada data yang di gunakan adalah desimal 100-359 sehingga dibutuhkan faktor koreksi pada counter R16 nya yaitu digunakan DAA agar digitnya tidak melebihi 9. Counter digit tengah dan digit kanan pun dibatasi sampai $60 sehingga data akhir nya yaitu 359. Pada program data pun, kode karakter yang dimasukan adalah pola karakter 0-9.

4. Pada program nomor 4 ini tidak jauh berbeda dengan program nomor 3 sebelumnya. Perbedaan adalah program counter ini dilakukan berulang-ulang dengan cara membuat label A pada awal program, lalu saat program telah selesai mencapai 359, program akan loncat lagi otomatis ke awal dengan rjmp A sehingga akan terjadi infinite loop atau berulang terus menerus.

5. Pada program ini, akan dibuat sebuah program up-counter bilangan OKTAL dari 100 hingga 357. Proses dan konsepnya tidak jauh berbeda dengan program pada soal nomor sebelumnya. Perbedaan terletak pada digunakannya subrutin KOR yang akan mengoreksi bilangan sehingga tidak melebihi angka 7. Compaare hasil akhir pun

dibatasi sampai $60 sehingga nilai akhirnya adalah 357. Program area pun hanya dimasukan kode karakter 0-7.

6. Soal dan program pada nomor 6 ini sebenarnya mirip persis dengan noor sebelumnya. Cukup ditambahkan sebuah label baru ketika pengecekan data telah mencapai akhirnya, dimana perlu dibuat perbandingan juga untuk dapat melakukan lompatan ke label di awal program.

7 dan 8. Pada dua nomor ini, program yang hendak dibuat meupakan program up counter 5 bit dari data 00 – 1F dan up counter 8 bit dari 00 – FF. Operasi yang digunakan cukup sama dan mengacu pada nomor-nomor sebelumnya dengan beberapa perbedaan. Counter yang digunakan hanya 1 karena tidak adanya digit ketiga yang digunakan. Dilakukan program sama seperti sebelumnya yaitu pertama digit MSB dahulu ($2) lalu digit LSB paling kanan ($1), caranya cukup sama dengan mengatur digit tengah dan digit paling kanan seperti soal sebelumnya. Digunakan juga kembali kode karakter heksadesimal, sehingga kodenya dari 0 – F.

Perbedaan kedua nomor ini adalah batas counternya (R16). Pada nomor 7 karena data maksimalnya 1F, maka dilakukan compare hingga $20. Untuk nomor 8, karena datanya maksumumnya FF maka dicompare dengan $00.

9. Program nomor 9 ini sangat berbeda dengan program sebelumnya karena bukan sebuah program counter. Program ini menggunakan output Multiple 7 Segment 8 Digit Common Anode. Digunakan Port A dan Port B sebagai output, dimana Port A sebagai output kode posisi, dan port B sebagai output kode karakter.

Pada program area diatur posisinya yaitu kode posisi dilanjut ke kode karakter lalu dilanjut kode posisi selanjutnya lalu kode karakter selanjutnya dan seterusnya hingga semua kode posisi sudah terpakai, Pada program, dimasukan terlebih dahulu kode posisi $80 sehingga urutan pengisiannya adalah dari kiri ke kanan. Pada program area, digunakan 2 label DAT dan DAT2 dimana DAT sebagai penyimpan kode posisi dan kode karakter “belajar”, dan DAT2 digunakan sebagai kode posisi dan kode karakter “rajinlah”. Program akan mencetak kata “belajar” dahulu. Diberikan inisiasi counter frame R16 sebanyak 240. Ini dimaksudkan agar kata “belajar” akan terus mencetak selama 4 detik sebelum ke proses selanjutnya. Untuk mengambil karakter “belajar” terlebih dahulu, maka lakukan load data dari program area DAT dahulu ke pointer z. Digunakan counter sebanyak 8 kali karena terdapat 8 posisi dan 8 karakter. Dilakukan pengoutputan kode posisi dahulu lewat load data program memory ke r21 lalu dioutputkan ke port A. Dilanjutkan dengan pengoutputan kode karakter lewat load data program memory ke r22 lalu dioutputkan ke port B. Proses ini akan berlangsung 8 kali sampai semua kode karakter dari “belajar” sudah tercetak. Ketika sudah mencapai

angka 8 ini, program akan terus mengulang hingga counter R16 (240) terus berkurang hingga 0. Setelah proses tersebut, program akan mereset seluruh 7 segment dengan memberi output port b dengan FF. Setelah clear, lakukan proses yang sama seperti sebelumnya hanya saja dengan kata “rajinlah”, maka dari itu digunakan label DAT2 untuk memanggil kode karakter “rajinlah”, selebihnya prosesnya sama. Pada semua kode karakter “rajinlah” sudah muncul dan counter frame nya sudah habis, loncatkan kembali ke awal program dengan **RJMP HERE** agar data muncul berulang-ulang.

10. Program no 10 ini tidak jauh berbeda dengan program pada soal nomor 9. Terdapat 3 kata yaitu “—rajin-“, “—tulus-“, dan “—jujur-“ sehingga diperlukan 3 label pada program memory yairu DAT,DAT1, dan DAT2. Proses nya cukup sama yaitu dengan menggunakan counter frame r16 240 kali tiap katanya lalu dipanggila kode posisi dan kode karakternya.Karena terdapat 3 kata, maka proses ini menjadi 3 kali. Tidak lupa juga untuk selalu melakukan clean display dengan memberi nilai FF yang dioutputkan ke port b. Karena katanya berbeda dengan nomor sebelumnya, dibuat terlebih dahulu pola karakternya yang baru.

12. Program nomor 12 ini tidak berbeda jauh dengan program no 10 sebelumnya. Perbedaan adalah pada kata “—jujur-“ harus berkedip sekitar 2 kali tiap detik. Pada saat proses kata “—jujur-“ dimulai, dimasukan counter r18 sebanyak 8 kali untuk berkedip 2 kali tiap detik. Counter R16 diisi sebanyak 30, ini berfungsi sebagai counter saat 7 segmentnya mati karena berkedip, agar tidak terlalu lama maka hanya diberi counter sebanyak 30. Diambil kode posisi dan kode karakter kembali sama seperti nomor 10. Pada saat clear display, diberikan delay agar clear display ini tidak terlalu cepat, dilanjutkan dengan pengurangan counter r18 dan memunculkan kata jujur kembali juga dengan clear display nya. Ini membuat kata jujur berkedip. Untuk kata rajin dan tulus bisa disamakan dengan nomor 10.

VI. KESIMPULAN

Dari praktikum ini, dapat diambil kesimpulan bahwa untuk dapat mengatur fungsi dari port baik untuk input maupun output, perlu diperhatikan setiap penggunaan bit yang dibutuhkan untuk mengatur fungsi port itu sendiri baik sebagai input maupun output. Selain itu, untuk dapat menampilkan banyak digit 7-segment, dapat digunakan Multiple 7-segment 8 digit, dimana pada 7-segment ini, terdapat dua bagian pin input, satu untuk common (pengatur digit mana yang aktif) dan juga satu lagi adalah bagian pin digit 7-segment itu sendiri. Maka

dari itu, perlu diperhatikan pola karakter yang akan diinputkan pada multiple 7-segment ini dalam heksadesimal, dan jangan sampai tertukar dengan kombinasi untuk common.

VII. DAFTAR PUSTAKA

Yusrizal. 2016. *Mikrokontroler Atmega8535.* Yusrizal Weblog. (https://yusrizalandeslubs.wordpress.com/dasar-elektronika). Diakses 5 Oktober 2020