Mencari Data Terkecil dan Terbesar pada Atmega8535

Diperuntukkan untuk memenuhi salah satu tugas praktikum Mata Kuliah Aplikasi Mikrokontroler

Praktikum	: Aplikasi Mikrokontroler
Praktikum ke	: 4
Tanggal Praktikum	: Kamis, 22 Oktober 2020
Tanggal Pengumpulan Laporan	: Jum'at, 23 Oktober 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 penjumlahan desimal pada mikrokontroler Atmega8535
- Mahasiswa dapat memahami konsep faktor koreksi pada penjumlahan desimal

II. LANDASAN TEORI

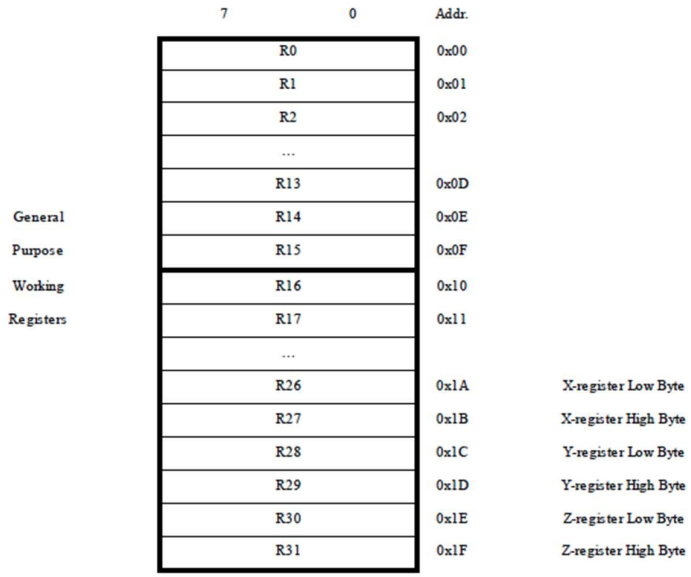
1. Mikrokontroler Atmega8535

   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 ADD dan ADC

   Penjumlahan pada Atmega8535 dilakukan dengan memanggil instruksi ADD, serta ADC untuk turut serta menambahkan carry yang terdapat pada flag dalam operasinya. Operand yang dapat digunakan dari kedua intruksi ini adalah R0 sampai dengan R31.

   Syntax = ADD Rd, Rr; Operasi = Rd ← Rd + Rr
   Syntax = ADC Rd, Rr; Operasi = Rd ← Rd + Rr + C

4. 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.

III. SOAL LAB

1. Buat program untuk mencari nilai terkecil dari 4 data biner 8-bit yang berada pada reg R6, R7, R8, R9. Simpan nilai data terkecil pada register R20.
2. Buat program untuk mencari nilai terkecil dari 4 data biner 16-bit yang berada pada reg R15:R14, R13:R12, R11:R10, R9:R8. Simpan nilai data terkecil pada register R20:R19.
3. Buat program untuk mencari nilai terbesar dari 16 data biner 8-bit yang berada pada reg R0 sampai R15. Simpan nilai data terbesar pada register R20.
4. Buat program untuk mencari nilai terbesar dari 16 data biner 8-bit yang berada pada program area dengan label DAT. Simpan nilai data terbesar pada register R20.
5. Buat program untuk mencari nilai terkecil dari 10 data biner 8-bit yang berada pada memori area dengan alamat awal $70. Simpan nilai data terkecil pada register R20.
6. Buat program untuk mencari nilai terkecil dari 10 data biner 16-bit yang berada pada program area dengan label DAT. Simpan nilai data terkecil pada register R20:R19.
7. Buat program untuk mencari nilai terkecil dari 10 data biner 16-bit yang berada pada unit memori dengan alamat awal $60. Simpan nilai data terkecil pada register R20:R19.
8. Buat program untuk menjumlahkan 4 data 7-bit yang tersimpan pada program area dengan label awal DAT1 dengan 4 data 7-bit lain pada program area dengan label DAT2. Simpan hasil penjumlahan pada memori unit mulai alamat $60.

IV. PROGRAM & HASIL

1. Program:

```
.INCLUDE "M8535DEF.INC"
.ORG 0
//AMIR HUSEIN (181344003)
RJMP MAIN
MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

MOV R20, R6
MOV R21, R7
RCALL PINDAH
MOV R21, R8
RCALL PINDAH
MOV R21, R9
RCALL PINDAH

AKHIR: RJMP AKHIR

PINDAH:
CP R20, R21
BRLO BACK
MOV R20, R21
BACK: RET
```

Hasil:

* R6 = 9C; R7 = F5; R8 = 9B; R9 = AA

```
R04= 0x00  R05= 0x00
R06= 0x9C  R07= 0xF5
R08= 0x9B  R09= 0xAA
R10= 0x00  R11= 0x00
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x00
R20= 0x9B  R21= 0xAA
R22= 0x00  R23= 0x00
```

* R6 = AA; R7 = 18; R8 = E1; R9 = C0

```
R04= 0x00  R05= 0x00
R06= 0xAA  R07= 0x18
R08= 0xE1  R09= 0xC0
R10= 0x00  R11= 0x00
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x00
R20= 0x18  R21= 0xC0
R22= 0x00  R23= 0x00
```

2. Program:

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

//INPUT (R15:R14)(R13:R12)(R11:R10)(R9:R8)
//OUTPUT (R20:R19)

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

//MSB
MOV R22, R15
MOV R21, R13
RCALL CYK
MOV R21, R11
RCALL CYK
MOV R21, R9
RCALL CYK

//PINDAH
MOV R20, R22

//LSB
MOV R22, R14
MOV R21, R12
RCALL CYK
MOV R21, R10
RCALL CYK
MOV R21, R8
RCALL CYK
```

```
//PINDAH
MOV R19, R22

AKHIR: RJMP AKHIR

CYK:
CP R22, R21
BRSH CONV
TER: RET

CONV:
MOV R22, R21
RJMP TER
```

Hasil:

* R15:R14 = 4AFF; R13:R12 = FFFE, R11:R10 = 2A43, R9:R8 = 4AFE

```
R06= 0x00  R07= 0x00
R08= 0xFE  R09= 0x4A
R10= 0x43  R11= 0x2A
R12= 0xFE  R13= 0xFF
R14= 0xFF  R15= 0x4A
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x43
R20= 0x2A  R21= 0xFE
R22= 0x43  R23= 0x00
R24= 0x00  R25= 0x00
```

3. Program:

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

RJMP MAIN
MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16
//INISIALISASI POINTER
LDI R30, 0X0001
LDI R31, 0

MOV R20, R0
RCALL CYB
AKHIR: RJMP AKHIR

CYB:
LD R25, Z+
CP R20, R25
BRSH SKIP
MOV R20, R25
SKIP:
CPI R30, $0F
BREQ AKHIR
RJMP CYB
```

Hasil:

```
* R00= 0xDD  R01= 0xA4
  R02= 0x55  R03= 0x70
  R04= 0x7F  R05= 0xE6
  R06= 0xD1  R07= 0xCD
  R08= 0xC9  R09= 0xC0
  R10= 0x0C  R11= 0x01
  R12= 0x02  R13= 0x0A
  R14= 0x6A  R15= 0x32
  R16= 0x02  R17= 0x00
  R18= 0x00  R19= 0x00
  R20= 0xE6  R21= 0x00
```

```
* R00= 0x32  R01= 0x53
  R02= 0x64  R03= 0x82
  R04= 0xBB  R05= 0xA6
  R06= 0x0C  R07= 0x54
  R08= 0x91  R09= 0x79
  R10= 0x89  R11= 0x99
  R12= 0x90  R13= 0x16
  R14= 0x00  R15= 0xAE
  R16= 0x02  R17= 0x00
  R18= 0x00  R19= 0x00
  R20= 0xBB  R21= 0x00
```

4. Program:

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

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

//INISIALISASI POINTER
LDI ZL, LOW(2*DAT)
LDI ZH, HIGH(2*DAT)
//INISIALISASI COUNTER
LDI R19,0
LPM R20, Z+
RCALL OPR

AKHIR: RJMP AKHIR

OPR:
LPM R21, Z+
CP R20, R21
BRSH SKIP
MOV R20, R21
SKIP:
INC R19
CPI R19, $0F
BREQ AKHIR
RJMP OPR

.ORG $80
DAT: .DB $43,$22,$F3,$A4,$9C,$36,$37,$34,$2A,$10,$11,$12,$CC,$A1,$E1
```

Hasil:

* Input = 43; 22; F3; A4; 9C; 36; 37; 34; 2A; 10; 11; 12; CC; A1; E1

```
R10= 0x00  R11= 0x00
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x0F
R20= 0xF3  R21= 0x00
R22= 0x00  R23= 0x00
R24= 0x00  R25= 0x00
R26= 0x00  R27= 0x00
```

* Input = 43; 4A; 3A; 2A; 11; 4D; 8C; 9A; 9E; 77; 76; 75; 12; 13; 01

```
R10= 0x00  R11= 0x00
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x0F
R20= 0x9E  R21= 0x00
R22= 0x00  R23= 0x00
R24= 0x00  R25= 0x00
R26= 0x00  R27= 0x00
```

5. Program:

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

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

//INISIALISASI POINTER
LDI ZL, LOW($70)
LDI ZH, HIGH($70)
//INISIALISASI COUNTER
LDI R19,0
LD R20, Z+
RCALL OPR

AKHIR: RJMP AKHIR

OPR:
LD R21, Z+
CP R20, R21
BRLO SKIP
MOV R20, R21
SKIP:
INC R19
CPI R19, $0A
BREQ AKHIR
RJMP OPR
```

Hasil:

*

Memory

Data | 8/16 | abc. | Address: 0x60

```
000060 75 02 FF FF FF FF  u.ÿÿÿÿ
000066 FF FF FF FF FF FF  ÿÿÿÿÿÿ
00006C FF FF FF FF 43 84  ÿÿÿÿC„
000072 85 86 80 AA AF 7E  …†€ª¯~
000078 9B 4E FF FF FF FF  ›Nÿÿÿÿ
00007E FF FF FF FF FF FF  ÿÿÿÿÿÿ
000084 FF FF FF FF FF FF  ÿÿÿÿÿÿ
00008A FF FF FF FF FF FF  ÿÿÿÿÿÿ
000090 FF FF FF FF FF FF  ÿÿÿÿÿÿ
```

```
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x0A
R20= 0x43  R21= 0xFF
R22= 0x00  R23= 0x00
```

*

Memory

Data | 8/16 | abc. | Address: 0x60

```
000060 75 02 FF FF FF FF  u.ÿÿÿÿ
000066 FF FF FF FF FF FF  ÿÿÿÿÿÿ
00006C FF FF FF FF 75 FF  ÿÿÿÿuÿ
000072 32 96 09 FE CF FC  2–.þÏü
000078 16 32 FF FF FF FF  .2ÿÿÿÿ
00007E FF FF FF FF FF FF  ÿÿÿÿÿÿ
000084 FF FF FF FF FF FF  ÿÿÿÿÿÿ
00008A FF FF FF FF FF FF  ÿÿÿÿÿÿ
000090 FF FF FF FF FF FF  ÿÿÿÿÿÿ
```

```
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x0A
R20= 0x09  R21= 0xFF
R22= 0x00  R23= 0x00
```

6. Program:

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

//INISIALISASI POINTER
LDI ZL, LOW(2*DAT)
LDI ZH, HIGH(2*DAT)
//INISIALISASI COUNTER
LDI R18,0
LPM R19, Z+
LPM R20, Z+
RCALL OPR

AKHIR: RJMP AKHIR

OPR:
CPI R18, $0A
BREQ AKHIR
INC R18
LPM R21, Z+
LPM R22, Z+
//CEK PULUHAN
CP R20, R22
BREQ CEKSAT
BRLO OPR
MOV R20, R22
MOV R19, R21
RJMP OPR

CEKSAT:
CP R19, R21
BRLO OPR
MOV R19, R21
RJMP OPR

.ORG $80
DAT: .DW $4300,$2200,$F303,$A454,$9C63,$3653,$10F0,$3424,$2AFF,$10F4,$11AF,$12B2,$CC3D,$A14B,$E1E3
```

Hasil:

* Input seperti pada program

```
R16= 0x02  R17= 0x00
R18= 0x0A  R19= 0xF0
R20= 0x10  R21= 0xAF
R22= 0x11  R23= 0x00
```

* Input =

```
DAT: .DW $64F0,$2210,$53FC,$1111,$932D,$9999,$1777,$1453,$8110,$A4DE,$184D,$999A,$43FC,$4A8B,$AAAA
```

```
R16= 0x02  R17= 0x00
R18= 0x0A  R19= 0x11
R20= 0x11  R21= 0x4D
R22= 0x18  R23= 0x00
```

7. Program: Hasil:

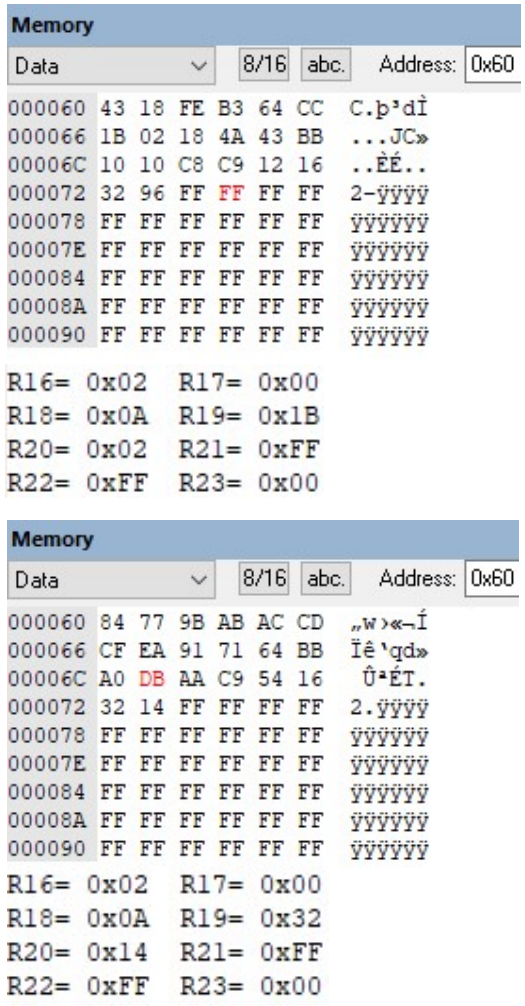
```
.INCLUDE "M8535DEF.INC"
.ORG 0
//AMIR HUSEIN
RJMP MAIN
MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

//INISIALISASI POINTER
LDI ZL, LOW($60)
LDI ZH, HIGH($60)
//INISIALISASI COUNTER
LDI R18,0
LD R19, Z+
LD R20, Z+
RCALL OPR

AKHIR: RJMP AKHIR

OPR:
CPI R18, $0A
BREQ AKHIR
INC R18
LD R21, Z+
LD R22, Z+
//CEK PULUHAN
CP R20, R22
BREQ CEKSAT
BRLO OPR
MOV R20, R22
MOV R19, R21
RJMP OPR

CEKSAT:
CP R19, R21
BRLO OPR
MOV R19, R21
RJMP OPR
```



8. Program:

```
.INCLUDE "M8535DEF.INC"
.ORG 0
//AMIR HUSEIN (181344003)
RJMP MAIN
MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

//POINTER INITIALIZATION
LDI R18, 1
//PROCESS DAT1 FIRST
LDI ZL, LOW(2*DAT1)
LDI ZH, HIGH(2*DAT1)

LPM R20, Z+
RCALL JUMLAH

//THEN PROCESS DAT2
LDI R18, 0
LDI ZL, LOW(2*DAT2)
LDI ZH, HIGH(2*DAT2)
RCALL JUMLAH

//STORE RESULT TO DATA AREA
LDI ZL, LOW(HASIL)
LDI ZH, HIGH(HASIL)
ST Z+, R20
ST Z+, R21

AKHIR: RJMP AKHIR

JUMLAH:
LPM R19, Z+
ADD R20, R19
ADC R21, R17
INC R18
CPI R18, $4
BRLO JUMLAH
RET

.ORG $90
DAT1: .DB $7F, $4A, $44, $5F

.ORG $96
DAT2: .DB $2D, $55, $77, $10

.ORG $60
HASIL: .DB 0
```

Hasil:

* Input =

```
.ORG $90
DAT1:  .DB $7F, $4A, $44, $5F

.ORG $96
DAT2:  .DB $2D, $55, $77, $10
```

Output =

Memory

Data 8/16 abc

```
000060 75 02 FF FF FF FF
000066 FF FF FF FF FF FF
00006C FF FF FF FF FF FF
000072 FF FF FF FF FF FF
000078 FF FF FF FF FF FF
00007E FF FF FF FF FF FF
000084 FF FF FF FF FF FF
00008A FF FF FF FF FF FF
000090 FF FF FF FF FF FF
```

* Input =

```
.ORG $90
DAT1:  .DB $1E, $32, $7A, $77

.ORG $96
DAT2:  .DB $54, $68, $69, $40
```

Output =

Memory

Data 8/16 abc

```
000060 A6 02 FF FF FF FF
000066 FF FF FF FF FF FF
00006C FF FF FF FF FF FF
000072 FF FF FF FF FF FF
000078 FF FF FF FF FF FF
00007E FF FF FF FF FF FF
000084 FF FF FF FF FF FF
00008A FF FF FF FF FF FF
000090 FF FF FF FF FF FF
```

V. ANALISIS

1. Pada program ini, akan dicari nilai terkecil dari data 8-bit yang ada pada register R6 sampai R9. Langkah awal yang perlu dilakukan ialah memindahkan R6 ke R20, dimana R20 disini akan bertindak sebagai register operasi perbandingan dan akan menyimpan hasil akhirnya juga nanti. R21 akan diisi oleh R7, R8 dan R9 berturut-turut dengan subrutin PINDAH, dimana didalamnya akan ada perbandingan instruksi CP antara R20 dan R21, bila R20 lebih kecil maka program akan RETURN, sedangkan bila tidak, R20 akan diisi oleh R21. Subrutin ini akan terus dipanggil sampai dengan memindahkan R9 sebagai register input yang terakhir.
2. Pada program ini, data yang diproses merupakan data 4 digit atau 16-bit biner. Konsep untuk menemukan nilai terkecil adalah dengan membandingkan 8-bit MSB dari semua data dahulu. Register yang akan digunakan untuk menampung data perbandingan sementara adalah register R22, dimana proses membandingkan data ada pada subrutin CYK yang akan dipanggil setiap kali ingin membandingkan data sesuai banyaknya data input. Setelah data 8-bit MSB dibandingkan, barulah membandingkan LSB nya.
3. Dalam menyelesaikan program ini, akan digunakan pointer Z sebagai penunjuk alamat register yang akan terus bertambah sampai alamat untuk R15. Batas peningkatan ini akan diatur oleh instruksi CPI pada register R30 yang ketika menyentuh nilai 0FH, maka program akan exit. Pada mulanya, data pada R0 akan dipindahkan ke R20, yang kemudian R20 ini akan dibandingkan dengan R25 yang akan menyimpan data pada register dengan bantuan perintah LD secara terus menerus. Ketika R20 < R25, maka R20 akan diisikan R25, sebaliknya bila tidak, maka program akan loncat ke label SKIP untuk R25 mengambil data dari register yang sudah di point oleh pointer Z.

4. Program ini sebenarnya sama dengan program pada soal sebelumnya yaitu nomor 3. Perbedaannya terletak pada data yang akan dibandingkan bukan dari register, melainkan dari program area dengan label DAT, dengan alamat awal $80. Pointer yang akan digunakan untuk menunjuk ke alamat program area adalah pointer Z. Register R19 akan bertindak sebagai counter untuk membatasi operasi sampai 15 kali. Register yang akan dibandingkan adalah R20 dnegan R21, dimana R21 nilainya akan berganti sesuai jumlah data yang ada. Ketika R20 >= R21, maka program akan loncat ke label SKIP untuk meningkatkan counter +1, sedangkan ketika R20 < R21, maka nilai R21 akan dipindahkan ke R20.
5. Pada program ini, konsep dalam membandingkan data sama seperti soal sebelumnya, apabila pada soal sebelumnya dicari nilai terbesar, maka pada soal ini dicari nilai terkecil. Data yang akan dicek juga berasal dari data area, sehingga diperlukan pointer Z sebagai penunjuk yang dibantu oleh perintah LD (Load Data).
6. Pada program ini, tetap digunakan pointer Z untuk menunjuk pada alamat program area berlabel DAT, dimana R18 bertindak sebagai counter. Hal yang perlu dilakukan dahulu ialah membandingkan nilai MSB 8-bit dahulu, dengan mengambil dari program area melalui instruksi LPM. R20 akan menampung nilai puluhan sementara, dan R19 akan menampung nilai satuan. R22 dan R21 adalah register yang akan terus diisi oleh data input untuk dibandingkan dengan R20:R19. R20 akan dibandingkan dahulu dengan R22, ketika R20 lebih besar, maka nilainya diisi oleh R22 yang kemudian otomatis R19 diisi oleh R21. Apabila nilainya lebih kecil maka program akan mengulang mengisi data ke R22:R21, sedangkan bila R20==R22, akan dilakukan cek satuan pada R19 dengan R21.
7. Program ini persis sama dengan program sebelumnya pada nomor 6. Perbedaannya ada pada data input yang berasal dari data area dengan alamat awal $60, sehingga pointer Z perlu diarahkan pada alamat tersebut, serta proses pengambilan data memanfaatkan instruksi LD, bukan LPM. Secara konsep perbandingan dan lainnya, sama seperti soal sebelumnya.
8. Pada program ini, akan dilakukan proses penjumlahan data pada program area berlabel DAT1 dan DAT2 yang masing-masing memiliki 4 data 7-bit biner, sehingga data terbesar yang memungkinkan adalah 7FH. Penjumlahan yang akan diproses terlebih dahulu adalah data yang ada pada DAT1, sehingga alamatnya ditunjuk diawal oleh pointer Z. Proses penjumlahan ini ada pada subrutin JUMLAH, dimana R21 menyimpan carry dari penjumlahan, dan R20 menyimpan 4-bit LSB, R19 akan selalu mengambil data dari pointer Z, dan dijumlahkan dengan R20. Setelah proses penjumlahan pada DAT1 selesai, barulah pointer Z akan menunjuk alamat DAT2 untuk diambil datanya dan kemudian dijumlahkan dengan data hasil penjumlahan DAT1. R18 akan bertindak sebagai counter, dimana saat memproses DAT1, R18 diisi 1, karena R20 sudah diisi data dari DAT1 diluar subrutin JUMLAH, dan saat memproses DAT2, R18 diinisialisasi 0. R18 ini akan dicek dengan $4, ketika nilainya telah menyentuh 4, maka program akan dikembalikan ke MAIN.

VI. KESIMPULAN

Dari praktikum LAB-4 yang telah dilaksanakan ini, dapat diambil kesimpulan bahwa dalam mencari nilai terkecil maupun terbesar pada data yang ada, dapat memanfaatkan instruksi perbandingan. Selain itu, apabila terdapat data 16-bit, maka perlu dibandingkan dahulu nilai 8-bit MSB, kemudian diikuti oleh 8-bit LSB. Pemanfaatan pointer juga perlu dilakukan apabila data input berada pada Data Area atau Program Area.

VII. DAFTAR PUSTAKA

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