

# Pengurangan Heksadesimal dan Penggunaan Alamat Memori

Diperuntukkan untuk memenuhi salah satu tugas praktikum Mata Kuliah Aplikasi Mikrokontroler



Praktikum	: Aplikasi Mikrokontroler
Praktikum ke	: 2
Tanggal Praktikum	: Kamis, 8 Oktober 2020
Tanggal Pengumpulan Laporan	: Selasa, 13 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 pengurangan pada mikrokontroler Atmega8535
- Mahasiswa dapat memahami prinsip pengurangan dengan memanfaatkan lokasi data area pada mikrokontroler Atmega8535

## II. LANDASAN TEORI

### 1. Mikrokontroler Atmega8535

Mikrokontroler merupakan suatu chip komputer mini, dimana di dalamnya sudah terdapat sebuah mikrosesor 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.

	7	0	Addr.	
General Purpose Working Registers	R0		0x00	
	R1		0x01	
	R2		0x02	
	...			
	R13		0x0D	
	R14		0x0E	
	R15		0x0F	
	R16		0x10	
	R17		0x11	
	...			
	R26		0x1A	X-register Low Byte
	R27		0x1B	X-register High Byte
	R28		0x1C	Y-register Low Byte
	R29		0x1D	Y-register High Byte
	R30		0x1E	Z-register Low Byte
	R31		0x1F	Z-register High Byte

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 SUB dan SBC

Operasi pengurangan pada Atmega8535 dapat dilakukan dengan menggunakan perintah SUB. Operasi pengurangan heksadesimal dengan menggunakan instruksi SUB tidak melibatkan carry (borrow) di dalamnya, dan hanya bisa dilakukan diantara 2 operand berupa register.

Operasi:  $Rd \leftarrow Rd - Rr$ ;  
Syntax: SUB Rd, Rr;  
Operands:  $0 \leq Rd, Rr \leq 31$

Sedangkan SBC digunakan untuk melakukan operasi pengurangan heksadesimal dengan melibatkan carry (borrow) yang terdapat pada register flag C, dan hanya bisa dilakukan diantara 2 operand berupa register.

Operasi:  $Rd \leftarrow Rd - Rr - C$ ;  
Syntax: SBC Rd, Rr;  
Operands:  $0 \leq Rd, Rr \leq 31$

### 4. Instruksi LPM

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 memori yang dideklarasikan.

## III. SOAL LAB

1. Buat program untuk mengurangkan data 8-bit yang ada pada register R7 dengan data 8-bit yang ada pada register R8. Simpan hasil pengurangan pada register R6.
2. Buat program untuk mengurangkan data 16-bit yang ada pada register R20:R19 dengan data 16-bit yang ada pada register R22:R21. Simpan hasil pengurangan pada register R24:R23.
3. Buat program untuk mengurangkan data 16-bit yang ada pada Area Memori [Data1] dengan data 16-bit yang ada pada Area Memori [Data2]. Simpan hasil pengurangan pada Area Memori [Hasil].
4. Buat program untuk mengurangkan data 32-bit yang ada pada Area Memori [Data1] dengan data 32-bit yang ada pada Area Memori [Data2]. Simpan hasil pengurangan pada Area Memori [Hasil].
5. Buat program untuk mengurangkan data 64-bit yang ada pada Area Memori [Data1] dengan data 64-bit yang ada pada Area Memori [Data2]. Simpan hasil pengurangan pada Area Memori [Hasil].
6. Buat program untuk mengurangkan data 32-bit yang ada pada Area Memori [SMBR] dengan data 24-bit yang ada pada Area Mmeori [Data1], [Data2], [Data3], [Data4]. [Data5]. Simpan hasil pengurangan pada Area Memori [Hasil].
7. Buat program untuk mengisi R0 sampai R31 dengan data heksadesimal FF. Selesaikan dengan 2 cara, yaitu tanpa menggunakan loop dan menggunakan loop.
8. Buat program untuk mengisi R0 sampai R31 dengan data heksadesimal 00 sampai 31. Selesaikan dengan 2 cara, yaitu tanpa menggunakan loop dan menggunakan loop.

9. Buat program untuk mengisi R0 sampai R31 dengan data heksadesimal 1F sampai 00. Selesaikan dengan 2 cara, yaitu tanpa menggunakan loop dan menggunakan loop.

#### IV. PROGRAM & HASIL

1. Program:

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

RJMP MAIN

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

MOV R6, R7
CP R8, R6
BRLO KURANG

AKHIR: RJMP AKHIR

KURANG:
SUB R6, R8
RET
```

Hasil:

\* 43- 3A = 09

R00= 0x00	R01= 0x00
R02= 0x00	R03= 0x00
R04= 0x00	R05= 0x00
R06= 0x09	R07= 0x43
R08= 0x3A	R09= 0x00

\* FF – 8E = 71

R00= 0x00	R01= 0x00
R02= 0x00	R03= 0x00
R04= 0x00	R05= 0x00
R06= 0x71	R07= 0xFF
R08= 0x8E	R09= 0x00

2. Program:

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

RJMP MAIN

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

MOV R23, R19
MOV R24, R20

CP R24, R22
BRLO AKHIR
SUB R23, R21
SUB R24, R22

AKHIR: RJMP AKHIR
```

Hasil:

\* FFFF – AEAE = 5151

R18= 0x00	R19= 0xFF
R20= 0xFF	R21= 0xAE
R22= 0xAE	R23= 0x51
R24= 0x51	R25= 0x00

\* ABAB – 1212 = 9999

R18= 0x00	R19= 0xAB
R20= 0xAB	R21= 0x12
R22= 0x12	R23= 0x99
R24= 0x99	R25= 0x00

3. Program:

```
.include "m8535def.inc"
.org 0

rjmp main

main:
LDI R17, LOW(RAMEND)
OUT SPL, R17
LDI R17, HIGH(RAMEND)
OUT SPH, R17

LDI ZL, LOW(2*DATA1)
LDI ZH, HIGH(2*DATA1)
//PINDAH KE R1:R2
LPM R2, Z+
LPM R1, Z+

LDI ZL, LOW(2*DATA2)
LDI ZH, HIGH(2*DATA2)
//R3:R4
LPM R4, Z+
LPM R3, Z+

SUB R2, R4
SBC R1, R3

AKHIR: RJMP AKHIR

.org 0x80
DATA1: .DW $F333
.org $0105
DATA2: .DW $21DC
.org 0x66
HASIL: .DW $0
```

Hasil:

\* F333 – 21DC = D157

Memory	
Data	8/16 abc
0000A8	FF FF FF FF FF FF
0000AE	FF FF FF FF FF FF
0000B4	FF FF FF FF FF FF
0000BA	FF FF FF FF FF FF
0000C0	FF FF FF FF FF FF
0000C6	FF FF FF FF FF FF
0000CC	57 D1 FF FF FF FF
0000D2	FF FF FF FF FF FF
0000D8	FF FF FF FF FF FF

#### 4. Program:

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

RJMP MAIN

MAIN: LDI R17,LOW(RAMEND)
      OUT SPL,R17
      LDI R17,HIGH(RAMEND)
      OUT SPH,R17

      LDI ZL,LOW(2*DATA1)
      LDI ZH,HIGH(2*DATA1)
      //GET DATA1 FROM MEMORY
      LPM R3,Z+
      LPM R2,Z+
      LPM R1,Z+
      LPM R0,Z+

      LDI ZL,LOW(2*DATA2)
      LDI ZH,HIGH(2*DATA2)
      //GET DATA2 FROM MEMORY
      LPM R7,Z+
      LPM R6,Z+
      LPM R5,Z+
      LPM R4,Z+

      //OPERASI PENGURANGAN
      SUB R3,R10
      SBC R2,R9
      SBC R1,R8
      SBC R0,R7

      LDI ZL,LOW(2*HASIL)
      LDI ZH,HIGH(2*HASIL)
      //STORE DATA KE HASIL
      ST Z+,R3
      ST Z+,R2
      ST Z+,R1
      ST Z+,R0

      AKHIR: RJMP AKHIR

      .ORG $0AA0
      DATA1: .DD $FEFEFEFE
      .ORG $0AA3
      DATA2: .DD $89ABCDEF
      .ORG $066
      HASIL: .DD $0
```

Hasil:

\* FEFEFEFE – 89ABCDEF = 7553310F

Memory	
Data	8/16
0000BA	FF FF FF FF FF FF
0000C0	FF FF FF FF FF FF
0000C6	FF FF FF FF FF FF
0000CC	0F 31 53 75 FF FF
0000D2	FF FF FF FF FF FF
0000D8	FF FF FF FF FF FF
0000DE	FF FF FF FF FF FF
0000E4	FF FF FF FF FF FF
0000EA	FF FF FF FF FF FF

#### 5. Program:

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

RJMP MAIN

MAIN : LDI R17,LOW(RAMEND)
      OUT SPL,R17
      LDI R17,HIGH(RAMEND)
      OUT SPH,R17

      //GET DATA1 FROM MEMORY
      LDI ZL,LOW(2*DATA1)
      LDI ZH,HIGH(2*DATA1)

      LPM R10,Z+
      LPM R9,Z+
      LPM R8,Z+
      LPM R7,Z+
      LPM R6,Z+
      LPM R5,Z+
      LPM R4,Z+
      LPM R3,Z+

      //STORE HASIL
      LDI ZL,LOW(2*HASIL)
      LDI ZH,HIGH(2*HASIL)
      ST Z+,R10
      ST Z+,R9
      ST Z+,R8
      ST Z+,R7
      ST Z+,R6
      ST Z+,R5
      ST Z+,R4
      ST Z+,R3

      AKHIR: RJMP AKHIR

      .ORG $0100
      DATA1: .DQ $FEFEFEFEFEFEFEFE
      .ORG $0105
      DATA2: .DQ $123456789ABCDEF0
      .ORG $066
      HASIL: .DQ $0
```

Hasil:

\*FEFEFEFEFEFEFEFE – 123456789ABCDEF0 =

Memory	
Data	8/16
0000BA	FF FF FF FF FF FF
0000C0	FF FF FF FF FF FF
0000C6	FF FF FF FF FF FF
0000CC	0E 20 42 64 86 A8
0000D2	CA EC FF FF FF FF
0000D8	FF FF FF FF FF FF
0000DE	FF FF FF FF FF FF
0000E4	FF FF FF FF FF FF
0000EA	FF FF FF FF FF FF

#### 6. Program:

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

RJMP MAIN

MAIN : LDI R17,LOW(RAMEND)
      OUT SPL,R17
      LDI R17,HIGH(RAMEND)
      OUT SPH,R17

      LDI ZL,LOW(2*SMBR)
      LDI ZH,HIGH(2*SMBR)
      LPM R3,Z+
      LPM R2,Z+
      LPM R1,Z+
      LPM R0,Z+

      LDI ZL,LOW(2*DATA1)
      LDI ZH,HIGH(2*DATA1)
      LPM R6,Z+
      LPM R5,Z+
      LPM R4,Z+

      SUB R3,R6
      SBC R2,R5
      SBC R1,R4
      SBC R0,R20

      LDI ZL,LOW(2*DATA2)
      LDI ZH,HIGH(2*DATA2)
      LPM R9,Z+
      LPM R8,Z+
      LPM R7,Z+

      SUB R3,R15
      SBC R2,R14
      SBC R1,R13
      SBC R0,R20

      LDI ZL,LOW(2*DATA3)
      LDI ZH,HIGH(2*DATA3)
      LPM R12,Z+
      LPM R11,Z+
      LPM R10,Z+

      SUB R3,R12
      SBC R2,R11
      SBC R1,R10
      SBC R0,R20

      LDI ZL,LOW(2*DATA4)
      LDI ZH,HIGH(2*DATA4)
      LPM R15,Z+
      LPM R14,Z+
      LPM R13,Z+

      SUB R3,R15
      SBC R2,R14
      SBC R1,R13
      SBC R0,R20

      LDI ZL,LOW(2*DATA5)
      LDI ZH,HIGH(2*DATA5)
      LPM R16,Z+
      LPM R15,Z+
      LPM R14,Z+

      SUB R3,R16
      SBC R2,R15
      SBC R1,R14
      SBC R0,R20

      AKHIR: RJMP AKHIR

      .ORG 0XFF0
      SMBR: .DD 0XFFFFFFFF
      .ORG 0XFF5
      DATA1: .DD 0X123456
      .ORG 0XFF8
      DATA2: .DD 0X112233
      .ORG 0XFFB
      DATA3: .DD 0X333333
      .ORG 0XFFE
      DATA4: .DD 0X222222
      .ORG 0XFE0
      DATA5: .DD 0X111111
```

Hasil:

\* FFFFFFFF – 123456 – 112233 – 333333 – 222222 – 111111 = FF764310

```
R00= 0xFF  R01= 0x76
R02= 0x43  R03= 0x10
R04= 0x12  R05= 0x34
R06= 0x56  R07= 0x11
```

7A. Program:

```
main : LDI R17, LOW(RAMEND)
        OUT SPL, R17
        LDI R17, HIGH(RAMEND)
        OUT SPH, R17
        LDI R31, 0x0FF
        MOV R0, R31
        MOV R1, R31
        MOV R2, R31
        MOV R3, R31
        MOV R4, R31
        MOV R5, R31
        MOV R6, R31
        MOV R7, R31
        MOV R8, R31
        MOV R9, R31
        MOV R10, R31
        MOV R11, R31
        MOV R12, R31
        MOV R13, R31
        MOV R14, R31
        MOV R15, R31
        MOV R16, R31
        MOV R17, R31
        MOV R18, R31
        MOV R19, R31
        MOV R20, R31
        MOV R21, R31
        MOV R22, R31
        MOV R23, R31
        MOV R24, R31
        MOV R25, R31
        MOV R26, R31
        MOV R27, R31
        MOV R28, R31
        MOV R29, R31
        MOV R30, R31
AKHIR : RJMP AKHIR
```

Hasil:

```
R00= 0xFF  R01= 0xFF
R02= 0xFF  R03= 0xFF
R04= 0xFF  R05= 0xFF
R06= 0xFF  R07= 0xFF
R08= 0xFF  R09= 0xFF
R10= 0xFF  R11= 0xFF
R12= 0xFF  R13= 0xFF
R14= 0xFF  R15= 0xFF
R16= 0xFF  R17= 0xFF
R18= 0xFF  R19= 0xFF
R20= 0xFF  R21= 0xFF
R22= 0xFF  R23= 0xFF
R24= 0xFF  R25= 0xFF
R26= 0xFF  R27= 0xFF
R28= 0xFF  R29= 0xFF
R30= 0xFF  R31= 0xFF
```

7B. Program:

```
.include "m8535def.inc"
.org 0
rjmp main
main : LDI R17, LOW(RAMEND)
        OUT SPL, R17
        LDI R17, HIGH(RAMEND)
        OUT SPH, R17
        LDI R29, 0x0FF
LOOPZ: ST Z+, R29
        CPI R30, $1D
        BRNE LOOPZ
        MOV R30, R29
        MOV R31, R29
AKHIR : RJMP AKHIR
```

Hasil:

```
R00= 0xFF  R01= 0xFF
R02= 0xFF  R03= 0xFF
R04= 0xFF  R05= 0xFF
R06= 0xFF  R07= 0xFF
R08= 0xFF  R09= 0xFF
R10= 0xFF  R11= 0xFF
R12= 0xFF  R13= 0xFF
R14= 0xFF  R15= 0xFF
R16= 0xFF  R17= 0xFF
R18= 0xFF  R19= 0xFF
R20= 0xFF  R21= 0xFF
R22= 0xFF  R23= 0xFF
R24= 0xFF  R25= 0xFF
R26= 0xFF  R27= 0xFF
R28= 0xFF  R29= 0xFF
R30= 0xFF  R31= 0xFF
```



### 8A. Program:

```
main : LDI R17,LOW(RAMEND)
      OUT SPL,R17
      LDI R17,HIGH(RAMEND)
      OUT SPH,R17
      LDI R31,0x1F
      DEC R31
      MOV R30,R31
      DEC R31
      MOV R29,R31
      DEC R31
      MOV R28,R31
      DEC R31
      MOV R27,R31
      DEC R31
      MOV R26,R31
      DEC R31
      MOV R25,R31
      DEC R31
      MOV R24,R31
      DEC R31
      MOV R23,R31
      DEC R31
      MOV R22,R31
      DEC R31
      MOV R21,R31
      DEC R31
      MOV R20,R31
      DEC R31
      MOV R19,R31
      DEC R31
      MOV R18,R31
      DEC R31
      MOV R17,R31
      DEC R31
      MOV R16,R31
      DEC R31
      MOV R15,R31
      DEC R31
      MOV R14,R31
      DEC R31
      MOV R13,R31
      DEC R31
      MOV R12,R31
      DEC R31
      MOV R11,R31
      DEC R31
      MOV R10,R31
      DEC R31
      MOV R9,R31
      DEC R31
      MOV R8,R31
      DEC R31
      MOV R7,R31
      DEC R31
      MOV R6,R31
      DEC R31
      MOV R5,R31
      DEC R31
      MOV R4,R31
      DEC R31
      MOV R3,R31
      DEC R31
      MOV R2,R31
      DEC R31
      MOV R1,R31
      DEC R31
      MOV R0,R31
      LDI R31,0x1F
      AKHIR : RJMP AKHIR
```

### Hasil:

```
R00= 0x00 R01= 0x01
R02= 0x02 R03= 0x03
R04= 0x04 R05= 0x05
R06= 0x06 R07= 0x07
R08= 0x08 R09= 0x09
R10= 0x0A R11= 0x0B
R12= 0x0C R13= 0x0D
R14= 0x0E R15= 0x0F
R16= 0x10 R17= 0x11
R18= 0x12 R19= 0x13
R20= 0x14 R21= 0x15
R22= 0x16 R23= 0x17
R24= 0x18 R25= 0x19
R26= 0x1A R27= 0x1B
R28= 0x1C R29= 0x1D
R30= 0x1E R31= 0x1F
```

### 8B. Program:

```
main : LDI R17,LOW(RAMEND)
      OUT SPL,R17
      LDI R17,HIGH(RAMEND)
      OUT SPH,R17
      LDI R29,0
      ST Z+,R29
      INC R29
      CPI R30,$1D
      BRNE ULANG
      LDI R30,$1E
      LDI R31,$1F
      AKHIR : RJMP AKHIR
```

### Hasil:

```
R00= 0x00 R01= 0x01
R02= 0x02 R03= 0x03
R04= 0x04 R05= 0x05
R06= 0x06 R07= 0x07
R08= 0x08 R09= 0x09
R10= 0x0A R11= 0x0B
R12= 0x0C R13= 0x0D
R14= 0x0E R15= 0x0F
R16= 0x10 R17= 0x11
R18= 0x12 R19= 0x13
R20= 0x14 R21= 0x15
R22= 0x16 R23= 0x17
R24= 0x18 R25= 0x19
R26= 0x1A R27= 0x1B
R28= 0x1C R29= 0x1D
R30= 0x1E R31= 0x1F
```

### 9A. Program:

```
main : LDI R17,LOW(RAMEND)      MOV R16,R31
      OUT SPL,R17              INC R31
      LDI R17,HIGH(RAMEND)     MOV R15,R31
      OUT SPH,R17             INC R31
      INC R31                 MOV R14,R31
      MOV R30,R31             INC R31
      INC R31                 MOV R13,R31
      MOV R29,R31             INC R31
      INC R31                 MOV R12,R31
      MOV R28,R31             INC R31
      INC R31                 MOV R11,R31
      MOV R27,R31             INC R31
      INC R31                 MOV R10,R31
      MOV R26,R31             INC R31
      INC R31                 MOV R9,R31
      MOV R25,R31             INC R31
      INC R31                 MOV R8,R31
      MOV R24,R31             INC R31
      INC R31                 MOV R7,R31
      MOV R23,R31             INC R31
      INC R31                 MOV R6,R31
      MOV R22,R31             INC R31
      INC R31                 MOV R5,R31
      MOV R21,R31             INC R31
      INC R31                 MOV R4,R31
      MOV R20,R31             INC R31
      INC R31                 MOV R3,R31
      MOV R19,R31             INC R31
      INC R31                 MOV R2,R31
      MOV R18,R31             INC R31
      INC R31                 MOV R1,R31
      MOV R17,R31             INC R31
      INC R31                 MOV R0,R31
      LDI R31,0               LDI R31,0
      AKHIR : RJMP AKHIR
```

### Hasil:

```
R00= 0x1F  R01= 0x1E
R02= 0x1D  R03= 0x1C
R04= 0x1B  R05= 0x1A
R06= 0x19  R07= 0x18
R08= 0x17  R09= 0x16
R10= 0x15  R11= 0x14
R12= 0x13  R13= 0x12
R14= 0x11  R15= 0x10
R16= 0x0F  R17= 0x0E
R18= 0x0D  R19= 0x0C
R20= 0x0B  R21= 0x0A
R22= 0x09  R23= 0x08
R24= 0x07  R25= 0x06
R26= 0x05  R27= 0x04
R28= 0x03  R29= 0x02
R30= 0x01  R31= 0x00
```

### 9B. Program:

```
main : LDI R17,LOW(RAMEND)
      OUT SPL,R17
      LDI R17,HIGH(RAMEND)
      OUT SPH,R17
      LDI R29,0x1F
      ULANG: ST Z+,R29
      DEC R29
      CPI R30,$1D
      BRNE ULANG
      LDI R30,$1
      LDI R31,$0
      AKHIR : RJMP AKHIR
```

### Hasil:

```
R00= 0x1F  R01= 0x1E
R02= 0x1D  R03= 0x1C
R04= 0x1B  R05= 0x1A
R06= 0x19  R07= 0x18
R08= 0x17  R09= 0x16
R10= 0x15  R11= 0x14
R12= 0x13  R13= 0x12
R14= 0x11  R15= 0x10
R16= 0x0F  R17= 0x0E
R18= 0x0D  R19= 0x0C
R20= 0x0B  R21= 0x0A
R22= 0x09  R23= 0x08
R24= 0x07  R25= 0x06
R26= 0x05  R27= 0x04
R28= 0x03  R29= 0x02
R30= 0x01  R31= 0x00
```

## V. ANALISIS

1. Pada program ini, data pada R7 akan dikurangi dengan data pada R8, dan hasilnya akan disimpan di R6. Hal pertama yang perlu dilakukan ialah memindahkan dahulu data yang ada pada R7 ke register R6, lalu setelah itu data pada R8 akan dibandingkan dengan R6, bila  $R8 < R6$ , maka akan dilakukan operasi pengurangan dengan instruksi SUB.
2. Program ini hampir sama seperti pada soal sebelumnya, letak perbedaannya ada pada lebar data yang dilakukan pengurangan adalah 16-bit, sehingga diperlukan 2 register 8-bit untuk masing-masing data. Isi dari register R20:R19 dipindahkan dahulu ke register



target untuk hasil pengurangan yaitu R24:R23, lalu dilakukan pengecekan dahulu apakah  $R24 > R22$ , akan dilakukan operasi pengurangan SUB untuk 8-bit LSB dan SBC untuk 8-bit MSB.

3. Pada program ini, akan digunakan area memori untuk menyimpan data pengurangan dan hasilnya. Data pengurangan yaitu [DATA1] dan [DATA2] serta hasil [HASIL] perlu diinisialisasi dahulu, bisa diawal maupun diakhir. Tahapan inisialisasi ini diawali oleh menentukan alamat awal dengan Directives .ORG.

Setelah itu pada subrutin MAIN, akan dilakukan proses pointing atau penunjukkan oleh pointer Z (digunakan pointer Z karena mendukung pointing bagi data area dan program area) pada tiap data yang akan dikurangi serta terjadi juga proses pengambilan data dari alamat memori untuk dipindahkan ke register agar bisa dilakukan proses operasi pengurangan.

Pointer Z LOW akan diisi alamat dari 2 kali label data bagian LSB, begitupun dengan Z HIGH akan diisi alamat dari 2 kali label data bagian MSB nya. Pada kondisi ini, pointer Z sudah menunjuk ke alamat yang dituju, lalu dapat dilakukan pemindahan nilai ke register yang diinginkan dengan menggunakan instruksi LPM (Load Program Memory). Terlihat pada program diatas, R1:R2 akan diisi oleh data [DATA1], dan R4:R3 akan diisi oleh [DATA2], selanjutnya hanya tinggal dilakukan saja proses SUB dan SBC sesuai dengan letak tiap 8-bit data, setelah itu lakukan kembali proses pointing agar pointer Z menunjuk ke alamat [HASIL], dan dilakukan lah instruksi ST (Set), yang berfungsi untuk mengisi alamat dengan data yang berasal dari register. Pada hasilnya terlihat bahwa alamat awal yang menampung hasil bukanlah 0x66 seperti pada inisialisasi, akan tetapi pada alamat 0x0CC, karena lebar dari pengalamatan adalah 16-bit, sehingga 0x066 dikali 2 akan menghasilkan alamat 0x0CC pada Data Memory untuk LSB dan 0x0CD untuk MSB.

4. Program ini sebenarnya persis dengan program sebelumnya nomor 3. Letak perbedaannya hanya ada pada lebar data yang digunakan untuk operasi pengurangan adalah 32-bit, sehingga pada inisialisasi nilai [DATA1] dan [DATA2] digunakan .DD untuk bisa menampung nilai 32-bit.

Proses pengambilan nilai dari alamat yang sudah di pointing oleh pointer Z pun menjadi 4 kali, dikarenakan register GPR pada Atmega8535 bertipe 8-bit. Untuk hasil sendiri tetap sama yaitu dilakukan instruksi ST dimana hasil akhir akan disimpan pada alamat 0x0CC pada Data Memory untuk LSB dengan 8-bit MSB terletak pada alamat 0x0CF.

5. Program ini pun sama persis seperti sebelumnya, lebar data yang digunakan kali ini untuk proses pengurangan adalah 64-bit, sehingga digunakan .DQ saat inisialisasi agar bisa menampung 64-bit. Proses LPM tiap data pun menjadi 8 kali dengan kebutuhan 8 GPR, serta hasil akhir disimpan pada label [HASIL] yang terletak pada alamat 0x0CC untuk 8-bit LSB dan 0x0D3 untuk 8-bit MSB pada Data Memory.

6. Program ini terlihat seperti rumit, akan tetapi sebenarnya cukup mudah, data [DATA1] tetap diinisialisasi kan dahulu diawal, begitupun untuk data-data 24-bit lainnya yang akan menjadi pengurang. Proses pengurangan tetap dilakukan seperti biasa, hanya saja proses pengambilan nilai dari alamat program menjadi 6 kali, 1 untuk [DATA1] dan 5 untuk data pengurangnya. Setelah dipindah ke register, tinggal dilakukan pengurangan saja dengan SUB untuk 8-bit LSB dan sisanya menggunakan SBC.
7. A.) Pada program ini, dilakukan pengisian register tanpa looping, sehingga perlu dilakukan manual dengan mengisi terlebih dahulu R31 oleh 0x0FF dengan perintah LDI. Langkah selanjutnya adalah melakukan instruksi MOV dari R31 ke masing-masing register sampai R30.  
  
B.) Pada program ini, dilakukan pengisian register dengan looping, sehingga bisa meminimalisir waktu dan panjang kode. Register yang diisi terlebih dahulu adalah R29 oleh 0x0FF, dikarenakan R30 dan R31 digunakan sebagai pointer nilai register, dimana R0 merupakan alamat 0x0, R1 0x0001 dan seterusnya. Selanjutnya dilakukan perintah (ST Z+, R29), dimana nilai pointer Z akan meningkat terus yang diisi oleh 0x0FF dari R29. Proses looping ini akan berhenti sampai R29, kemudian R30 dan R31 akan dilakukan pengisian nilai secara manual, karena apabila terus dilanjutkan, nilai pointer Z akan berubah menjadi FF dan merusak tujuan awal program.
8. A.) Pada program ini, dilakukan pengisian manual, dengan register awal yang diisi adalah R31 oleh 0x1F, yang dimana R31 ini akan selalu di kurangi 1 dengan instruksi DEC, lalu dilakukan LDI untuk masing-masing register sampai R0 oleh R31.  
  
B.) Pada program ini, dilakukan metoda looping dengan inisialisasi nilai R29 awal adalah 0. Kemudian digunakan pointer Z sebagai penunjuk untuk alamat register, dan setiap kali selesai melakukan perintah ST, akan dilakukan INC untuk R29, agar nilainya meningkat terus. Proses looping ini akan berhenti di R29, lalu R30 dan R31 akan dilakukan pengisian manual.
9. A.) Pada program ini, dilakukan tanpa metoda loop, sehingga perlu satu persatu register diisi. R31 akan dilakukan peningkatan sebesar 1, setelah itu diisikan ke register R30 satu persatu sampai R0. Setelah itu, R31 diisi 0 oleh perintah LDI, karena bila tidak, R31 akan sama nilainya dengan R0.  
  
B.) Program ini sama persis seperti program pada 8B. Yang perlu diubah hanyalah nilai R29 awal menjadi 0x1F dan R29 akan dilakukan perintah DEC setiap kali selesai melakukan pengisian ke register dengan memanfaatkan register pointer Z.

## VI. DAFTAR PUSTAKA

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