Penjumlahan dan Perkalian dengan Metode Penggandaan

Diperuntukkan untuk memenuhi salah satu tugas praktikum Mata Kuliah Aplikasi Mikrokontroler

Praktikum	: Aplikasi Mikrokontroler
Praktikum ke	: 1
Tanggal Praktikum	: Kamis, 1 Oktober 2020
Tanggal Pengumpulan Laporan	: Senin, 5 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 pada mikrokontroler Atmega8535
- Mahasiswa dapat memahami prinsip perkalian dengan metoda penggandaan pada mikrokontroler Atmega8535

II. LANDASAN TEORI

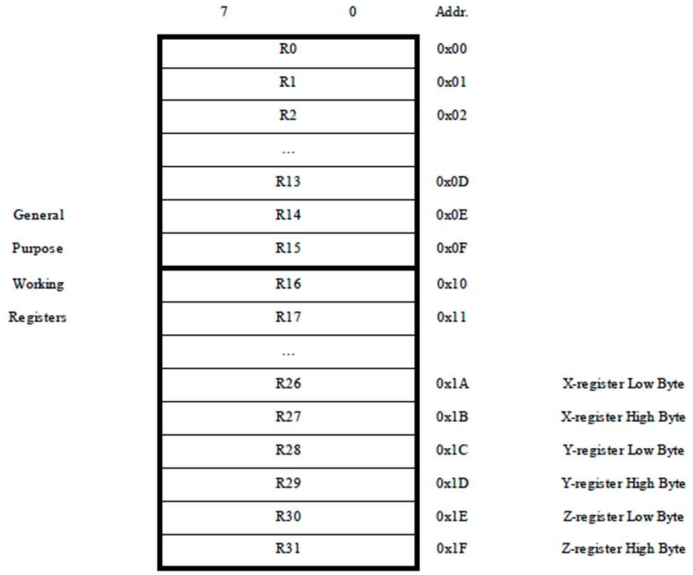
1. Mikrokontroler Atmega8535

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. Penjumlahan dan perkalian dengan metoda penggandaan

   Penjumlahan pada Atmega8535 dilakukan dengan memanggil instruksi ADD, serta ADC untuk turut serta menambahkan carry yang terdapat pada flag dalam operasinya. Selain itu, perkalian dengan metoda penggandaan dilakukan dengan melakukan penjumlahan dengan dirinya sendiri sebanyak n dari 2^n dimana hasil dari pangkat merupakan faktor pengali. Misal jika faktor pengali merupakan 48, maka faktor pengali tersebut dapat disusun sebagai berikut (32+16), berarti nilai akan digandakan sebanyak 5 kali lalu dijumlahkan dengan hasil faktor pengali 16.

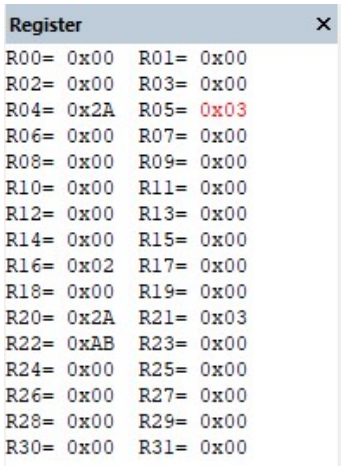
III. SOAL LAB

1. Buat program untuk menjumlahkan Data Heksadesimal: FF + 7D + 6A + 99 + AB. Simpan hasil penjumlahan pada register R5:R4.
2. Buat program untuk menjumlahkan Data Heksadesimal: FFFF + ABCD + 90AB + 567D + C4E2 + BEDA. Simpan hasil penjumlahan pada register R25:R24:R23.
3. Buat program untuk mengalikan Data Heksadesimal 2 digit yang ada pada register R16 dengan faktor pengali 16. Simpan hasil perkalian pada register R20:R19.
4. Buat program untuk mengalikan Data Heksadesimal 2 digit yang ada pada register R16 dengan faktor pengali 43. Simpan hasil perkalian pada register R20:R19.
5. Buat program untuk mengalikan Data Heksadesimal 4 digit yang ada pada register R17:R16 dengan faktor pengali $57. Simpan hasil perkalian pada register R20:R19:R18.
6. Buat program untuk mengalikan Data Heksadesimal 6 digit yang ada pada register R23:R22:R21 dengan faktor pengali $32. Simpan hasil perkalian pada register R28:R27:R26:R25.

IV. PROGRAM & HASIL

1.
```
.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
LDI R22, 0X7D
ADD R20, R22
ADC R21, R1
LDI R22, 0X6A
ADD R20, R22
ADC R21, R1
LDI R22, 0X99
ADD R20, R22
ADC R21, R1
LDI R22, 0X0AB
ADD R20, R22
ADC R21, R1

MOV R4, R20
MOV R5, R21

AKHIR: RJMP AKHIR
```

Hasil:

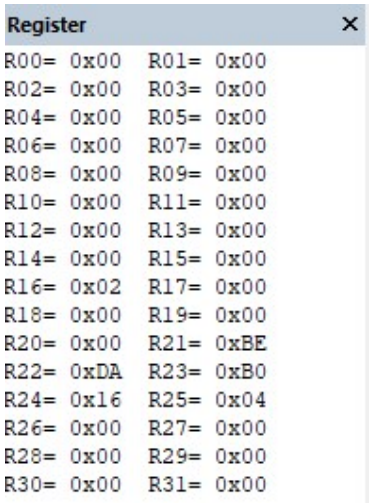


2. Program:

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

RJMP MAIN

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

LDI R23, 0X0FF
LDI R24, 0X0FF
LDI R22, 0X0CD
LDI R21, 0X0AB
RCALL OPP

LDI R22, 0X0AB
LDI R21, 0X90
RCALL OPP

LDI R22, 0X7D
LDI R21, 0X56
RCALL OPP

LDI R22, 0X0E2
LDI R21, 0X0C4
RCALL OPP

LDI R22, 0X0DA
LDI R21, 0X0BE
RCALL OPP

AKHIR: RJMP AKHIR
```

```
OPP:
ADD R23, R22
ADC R24, R21
ADC R25, R1
RET
```

Hasil:

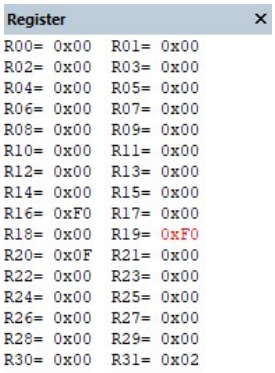


3. Program:

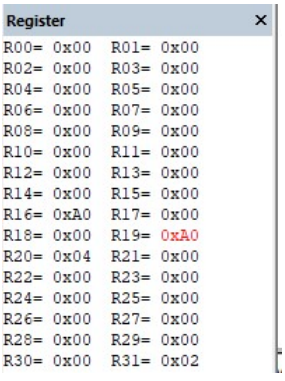
```
.INCLUDE "M8535DEF.INC"
.ORG 0

RJMP MAIN

MAIN:
LDI R31, LOW(RAMEND)
OUT SPL, R31
LDI R31, HIGH(RAMEND)
OUT SPH, R31

RCALL OPP
RCALL OPP
RCALL OPP
RCALL OPP

MOV R19, R16

AKHIR: RJMP AKHIR


OPP:
ADD R16, R16
ADC R20, R20
RET
```

Hasil:

- R16 = FF, R20:R19 = FF0



- R16 = 4A, R20:R19 = 4A0



4. Program:

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

RJMP MAIN

MAIN:
LDI R31, LOW(RAMEND)
OUT SPL, R31
LDI R31, HIGH(RAMEND)
OUT SPH, R31

MOV R2, R16

RCALL OPP
MOV R3, R16
MOV R4, R17
RCALL OPP
RCALL OPP
MOV R5, R16
MOV R6, R17
RCALL OPP
RCALL OPP

//JUMLAHKAN
ADD R16, R3
ADC R17, R4

ADD R16, R5
ADC R17, R6

ADD R16, R2
ADC R17, R1

//PINDAHKAN KE REGISTER TUJUAN
MOV R19, R16
MOV R20, R17

AKHIR: RJMP AKHIR
OPP:
ADD R16, R16
ADC R17, R17
RET
```

Hasil:

- R16 = FF, R20:R19 = 2AD5

Register

```
R00= 0x00  R01= 0x00
R02= 0xFF  R03= 0xFE
R04= 0x01  R05= 0xF8
R06= 0x07  R07= 0x00
R08= 0x00  R09= 0x00
R10= 0x00  R11= 0x00
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0xD5  R17= 0x2A
R18= 0x00  R19= 0xD5
R20= 0x2A  R21= 0x00
R22= 0x00  R23= 0x00
R24= 0x00  R25= 0x00
R26= 0x00  R27= 0x00
R28= 0x00  R29= 0x00
R30= 0x00  R31= 0x02
```

- R16 = E7, R20:R19 = 26CD

Register

```
R00= 0x00  R01= 0x00
R02= 0xE7  R03= 0xCE
R04= 0x01  R05= 0x38
R06= 0x07  R07= 0x00
R08= 0x00  R09= 0x00
R10= 0x00  R11= 0x00
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0xCD  R17= 0x26
R18= 0x00  R19= 0xCD
R20= 0x26  R21= 0x00
R22= 0x00  R23= 0x00
R24= 0x00  R25= 0x00
R26= 0x00  R27= 0x00
R28= 0x00  R29= 0x00
R30= 0x00  R31= 0x02
```

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 NILAI
ADD R18, R16
ADD R19, R17

//SIMPAN NILAI UNTUK PENJUMLAHAN TERAKHIR
ADD R13, R16
ADD R14, R17

//FAKTOR PENGALI 2
ADD R18, R16
ADC R19, R17
ADC R20, R1

MOV R1, R18
MOV R2, R19
MOV R3, R20

//FAKTOR PENGALI 4
RCALL FK

MOV R4, R18
MOV R5, R19
MOV R6, R20
```

Hasil:

- R17:R16 = FFFF, R20:R19:R18 = 57A8B0

Register

```
R00= 0x00  R01= 0x04
R02= 0xFE  R03= 0x01
R04= 0x08  R05= 0xFC
R06= 0x03  R07= 0x20
R08= 0xF0  R09= 0x0F
R10= 0x40  R11= 0xE0
R12= 0x1F  R13= 0x02
R14= 0xFF  R15= 0x00
R16= 0x02  R17= 0xFF
R18= 0xB0  R19= 0xA8
R20= 0x57  R21= 0x00
R22= 0x00  R23= 0x00
R24= 0x00  R25= 0x00
R26= 0x00  R27= 0x00
R28= 0x00  R29= 0x00
R30= 0x00  R31= 0x00
```

```
//FAKTOR PENGALI 8
RCALL FK
//FAKTOR PENGALI 16
RCALL FK

MOV R7, R18
MOV R8, R19
MOV R9, R20
//FAKTOR PENGALI 32
RCALL FK

ADD R10, R18
ADC R11, R19
ADC R12, R20
//FAKTOR PENGALI 64
RCALL FK

//MULAI PENJUMLAHAN
ADD R18, R7
ADC R19, R8
ADC R20, R9

ADD R18, R4
ADC R19, R5
ADC R20, R6

ADD R18, R1
ADC R19, R2
ADC R20, R3

ADD R18, R16
ADC R19, R17
ADC R20, R15

ADD R18, R13
ADC R19, R14
ADC R20, R0
akhir: rjmp akhir

FK:
ADD R18, R18
ADC R19, R19
ADC R20, R20
RET
```

- R17:R16 = AAAA, R20:R19:R18 = 3A70B0

Register

R00= 0x00	R01= 0x04
R02= 0x54	R03= 0x01
R04= 0x08	R05= 0xA8
R06= 0x02	R07= 0x20
R08= 0xA0	R09= 0x0A
R10= 0x40	R11= 0x40
R12= 0x15	R13= 0x02
R14= 0xAA	R15= 0x00
R16= 0x02	R17= 0xAA
R18= 0xB0	R19= 0x70
R20= 0x3A	R21= 0x00
R22= 0x00	R23= 0x00
R24= 0x00	R25= 0x00
R26= 0x00	R27= 0x00
R28= 0x00	R29= 0x00
R30= 0x00	R31= 0x00

6. Program:

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

rjmp main

main :  LDI R17,LOW(RAMEND)
        OUT SPL,R17
        LDI R17,HIGH(RAMEND)
        OUT SPH,R17
        //PINDAH INPUT KE OUTPUT
        MOV R25,R21
        MOV R26,R22
        MOV R27,R23

        RCALL GANDA //FK2

        MOV R0,R25
        MOV R1,R26
        MOV R2,R27
        MOV R3,R28

        RCALL GANDA //FK4
        RCALL GANDA //FK8

        RCALL GANDA //FK16
        MOV R4,R25
        MOV R5,R26
        MOV R6,R27
        MOV R7,R28

        RCALL GANDA //FK32

        ADD R25,R4
        ADC R26,R5
        ADC R27,R6
        ADC R28,R7


        ADD R25,R4
        ADC R26,R5
        ADC R27,R6
        ADC R28,R7

        ADD R25,R0
        ADC R26,R1
        ADC R27,R2
        ADC R28,R3

AKHIR : RJMP AKHIR

GANDA : ADD R25,R25
        ADC R26,R26
        ADC R27,R27
        ADC R28,R28
        RET
```

Hasil:

- R23:R22:R21 = FFFFFF, R28:R27:R26:R25 = 31FFFFCE

Register

R00= 0xFE	R01= 0xFF
R02= 0xFF	R03= 0x01
R04= 0xF0	R05= 0xFF
R06= 0xFF	R07= 0x0F
R08= 0x00	R09= 0x00
R10= 0x00	R11= 0x00
R12= 0x00	R13= 0x00
R14= 0x00	R15= 0x00
R16= 0x00	R17= 0x02
R18= 0x00	R19= 0x00
R20= 0x00	R21= 0xFF
R22= 0xFF	R23= 0xFF
R24= 0x00	R25= 0xCE
R26= 0xFF	R27= 0xFF
R28= 0x31	R29= 0x00
R30= 0x00	R31= 0x00

V. ANALISIS

1. Pada program ini, R22 bertugas sebagai register penyimpan nilai yang akan dijumlahkan. Register R21 dan R20 menyimpan hasil penjumlahan sementara, untuk register R21 akans selalu dilakukan operasi ADC dengan register kosong R1, sehingga R21 hanya akan menyimpan dan menjumlahkan hasil carry saja. R20 akan selalu dijumlahkan dengan R22 yang nilainya akan selalu berubah setelah operasi penjumlahan selesai, dan pada akhirnya, masing-masing isi dari register R21:R20 akan dipindahkan ke register R5:R4.

2. Pada program ini, register R21 dan R22 bertindak untuk menyimpan nilai yang akan menjadi penjumlah, sehingga register ini akan selalu berubah nilainya setelah dilakukan penjumlahan. Akan tetapi, 16-bit pertama yaitu FFFF akan disimpan di awal pada register R24:R23. Subrutin OPP akan melakukan operasi penjumlahan R23 dengan R22 yang kemudian apabila terdapat carry, akan turut serta dijumlahkan pada saat operasi penjumlahan R24 dan R21 berlangsung. Untuk register R25, hanya akan dilakukan perintah ADC dengan register kosong R1, karena hanya berfungsi untuk menyimpan carry yang didapat dari penjumlahan R24 dan R21.

3. Pada program ini, perlu dilakukan penggandaan sebanyak 4 kali untuk data 8-bit di register R16. Penggunaan subrutin sangat membantu dalam melakukan tugas ini, yaitu perlu dibuat operasi ADD R16 dengan dirinya sendiri, yang kemudian setiap ada carry yang keluar, ditampung pada sebuah register yaitu R20. Pada langkah akhir setelah dilakukan penggandaan 4 kali, data pada register R16 perlu dipindahkan ke register target yaitu R19, karena sesuai dengan permintaan soal bahwa hasil jawaban disimpan pada register R20:R19.

4. Program ini sama seperti program sebelumnya yaitu nomor 3, hanya saja faktor pengalinya adalah 43 yang terdiri dari faktor pengali 32, 8, 2, dan 1. Subrutin yang menampung operasi penggandaan perlu dibuat, yaitu dengan menjumlahkan R16 dan R17 berturut-turut dengan dirinya sendiri, untuk R17 digunakan instruksi ADC untuk serta menampung carry yang ada. Setelah penggandaan faktor pengali 32 selesai, maka hanya tinggal dilakukan penjumlahan dengan faktor pengali 8, 2, dan 1. Pada baris akhir, nilai dari R16 dipindahkan ke R19 dan dari R17 dipindahkan ke R20.

5. Pada program ini, sama seperti program sebelum-sebelumnya, yang menjadi pembeda adalah data yang akan digandakan merupakan 16-bit atau 4 digit heksadesimal. Faktor pengali adalah $57 atau dalam desimal adalah 87, sehingga faktor pengali 64, 16, 4, 2, dan 1 perlu disimpan. Setelah itu hanya tinggal dilakukan penjumlahan saja dari semua nilai faktor pengali ini.

6. Data heksadesimal 6 digit akan dikalikan dengan fakor pengali $32 atau bila dalam desimal adalah 50. 50 terdiri dari 32, 16 dan 2, maka hasil dari faktor pengali tersebutlah yang harus disimpan. Register R3:R2:R1:R0 akan menyimpan hasil dari faktor pengali 2, register R7:R6:R5:R4 akan menyimpan hasil dari faktor pengali 16, serta hasil dari faktor pengali 32 tetap akan disimpan pada register target yaitu R28:R27:R26:R25.

Pada program ini digunakan sebuah subrutin dengan nama GANDA, yang berfungsi untuk melakukan perintah penjumlahan dengan data itu sendiri, sehingga subrutin ini akan selalu dipanggil. Setelah nilai dari faktor pengali 2, 16 dan 32 disimpan, hal terakhir yang dilakukan adalah melakukan penjumlahan dari semua faktor pengali tersebut yang disimpan di register target.

VI. DAFTAR PUSTAKA

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