Penjumlahan Desimal pada Atmega8535

Diperuntukkan untuk memenuhi salah satu tugas praktikum Mata Kuliah Aplikasi Mikrokontroler

Praktikum	: Aplikasi Mikrokontroler
Praktikum ke	: 3
Tanggal Praktikum	: Kamis, 15 Oktober 2020
Tanggal Pengumpulan Laporan	: Jum'at, 16 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

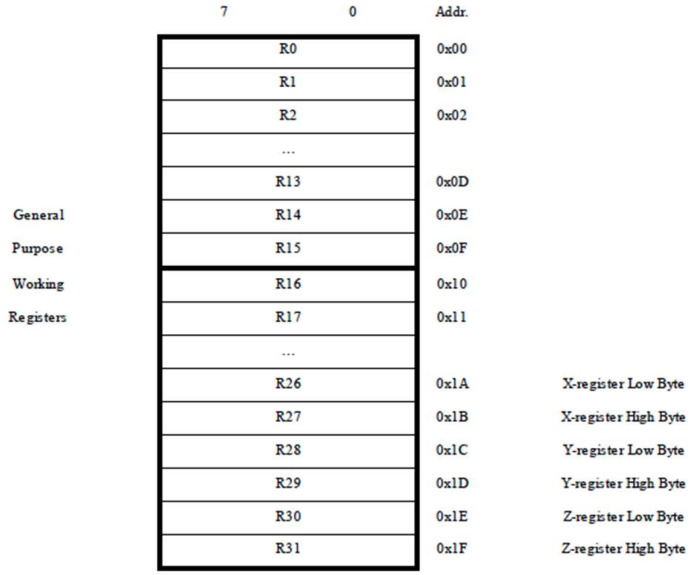
1. Mikrokontroler Atmega8535

   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

   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.

5. Faktor Koreksi

   Dalam melakukan penjumlahan desimal, komputer akan memproses data tersebut sebagai data biner serta melakukan seluruh kegiatan operasi dengan sistem biner. Untuk melakukan faktor koreksi ini, terlebih dahulu tiap 4-bit dicek, apabila nilainya melebihi 9, maka faktor koreksi dilakukan dengan melakukan penjumlahan dengan 6 atau 0110.

III. SOAL LAB

1. Buat program untuk menjumlahkan 2 data Desimal 2 DIGIT yang ada pada register R22 dan R23, simpan hasil penjumlahan pada Register R21:R20.
2. Buat program untuk menjumlahkan 10 data Desimal 2 DIGIT yang ada pada register R0 sampai R9, simpan hasil penjumlahan pada Register R21:R20.
3. Buat program untuk menjumlahkan 2 data Desimal 4 DIGIT yang ada pada register R21:R20 dan R23:R22, simpan hasil penjumlahan pada Register R26:R225:R24.
4. Buat program untuk menjumlahkan 10 data Desimal 4 DIGIT yang ada pada Program Area, simpan hasil penjumlahan pada Register R22:R21:R20.
5. Buat program untuk menjumlahkan 2 data Desimal 6 DIGIT yang ada pada Program area dengan Label var1 dan var2, simpan hasil penjumlahan pada Register R23:R22:R21:R20.
6. Buat program untuk menjumlahkan 2 data Desimal 8 DIGIT yang ada pada Program area dengan Label dat1 dan dat2, simpan hasil penjumlahan pada Register R24:R23:R22:R21:R20.
7. Buat program untuk menjumlahkan 2 data Desimal 10 DIGIT yang ada pada Program area dengan Label dat1 dan dat2, simpan hasil penjumlahan pada Register R25:R24:R23:R22:R21:R20.

IV. PROGRAM & HASIL

- **SUBRUTIN DAA**

  Pada Atmega8535, tidak terdapat instruksi DAA untuk melakukan koreksi seperti pada mikroprosesor 8086, karena DAA hanya ada pada mikroporsesor keluarga Intel saja. Maka dari itu, DAA ini perlu dirancang terlebih dahulu dan di *bundle* dalam bentuk subrutin. Subrutin DAA ini sudah disediakan oleh dosen penulis sekaligus pemrogram yaitu Pak Ferry, sehingga subrutin ini siap pakai dan diaplikasikan untuk melakukan koreksi. Program dari subrutin dapat dilihat dibawah ini:

```
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.
```

1. Program:

```
.include "m8535def.inc"
.org 0
RJMP MAIN
;Pemrogram : Amir Husein
;-------------------------
MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

MOV R19, R22

add r19,r23
rcall daa

mov r20,r19
ldi r21,0
adc r21,r21
AKHIR: RJMP AKHIR
```

Hasil:

* 99 + 99 = 198

```
R18= 0x00  R19= 0x98
R20= 0x98  R21= 0x01
R22= 0x99  R23= 0x99
R24= 0x00  R25= 0x00
```

* 84 + 39 = 123

```
R18= 0x00  R19= 0x23
R20= 0x23  R21= 0x01
R22= 0x84  R23= 0x39
R24= 0x00  R25= 0x00
```

2. Program:

```
.include "m8535def.inc"
.org 0
RJMP MAIN
;Pemrogram : Amir Husein
;-------------------------
MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

ULANG:
LD R28, Z+
ADD R19, R28
RCALL DAA
ADC R21, R10
CPI R30, $A
BRLO ULANG

MOV R20, R19

AKHIR: RJMP AKHIR
```

Hasil:

* 99+99+99+99+99+99+99+99+99+99 = 990

```
R00= 0x99  R01= 0x99
R02= 0x99  R03= 0x99
R04= 0x99  R05= 0x99
R06= 0x99  R07= 0x99
R08= 0x99  R09= 0x99
R10= 0x00  R11= 0x00
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x90
R20= 0x90  R21= 0x09
```

* 36+77+63+64+66+11+11+11+32+46 = 417

```
R00= 0x36  R01= 0x77
R02= 0x63  R03= 0x64
R04= 0x66  R05= 0x11
R06= 0x11  R07= 0x11
R08= 0x32  R09= 0x46
R10= 0x00  R11= 0x00
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x17
R20= 0x17  R21= 0x04
```

3. Program:

```
.include "m8535def.inc"
.org 0
RJMP MAIN
;Pemrogram : Amir Husein
;-------------------------------
MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

//PINDAH KE R19
MOV R19, R20
ADD R19, R22
RCALL DAA

//PINDAH HASIL DARI R19 KE R24
MOV R24, R19

//PINDAH KE R19
MOV R19, R21
ADC R19, R23
RCALL DAA

//PINDAH HASIL DARI 19 KE R25
MOV R25, R19

//ADC SISA UNTUK MSB
CLR R1
ADC R26, R1

AKHIR: RJMP AKHIR
```

Hasil:

* 6355 + 7777 = 14132

```
R20= 0x77  R21= 0x77
R22= 0x55  R23= 0x63
R24= 0x32  R25= 0x41
R26= 0x01  R27= 0x00
```

* 9999 + 9999 = 19998

```
R20= 0x99  R21= 0x99
R22= 0x99  R23= 0x99
R24= 0x98  R25= 0x99
R26= 0x01  R27= 0x00
```

* 1234 + 9876 = 11110

```
R20= 0x76  R21= 0x98
R22= 0x34  R23= 0x12
R24= 0x10  R25= 0x11
R26= 0x01  R27= 0x00
```

4. Program:

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

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

//INISIALISASI COUNTER
CLR R27

LPM R20,Z+
LPM R21,Z+

ULANG:
LPM R2,Z+
LPM R3,Z+

ADD R20,R2
MOV R19,R20
RCALL DAA
MOV R20,R19

ADC R21,R3
MOV R19,R21
RCALL DAA
MOV R21,R19

ADC R22,R1
INC R27
CPI R27, $9
BRNE ULANG

AKHIR: RJMP AKHIR

.ORG $80
DATA: .DW $9166, $9166, $9166, $9166, $9166, $9166, $9166, $9166, $9166, $9166
```

Hasil:

* 9166+9166+9166+9166+9166+9166+9166
9166+9166+9166 = 91660

```
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x16
R20= 0x60  R21= 0x16
R22= 0x09  R23= 0x00
```

5. Program:

```
.include "m8535def.inc"
.org 0
RJMP MAIN
;Pemrogram : Amir Husein
;-------------------------
MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

//GET DATA VAR1
LDI ZL, LOW(2*VAR1)
LDI ZH, HIGH(2*VAR1)
LPM R2, Z+
LPM R1, Z+
LPM R0, Z+

//GET DATA VAR2
LDI ZL, LOW(2*VAR2)
LDI ZH, HIGH(2*VAR2)
LPM R5, Z+
LPM R4, Z+
LPM R3, Z+

//PENJUMLAHAN
MOV R19, R2
ADD R19, R5
RCALL DAA
MOV R2, R19

MOV R19, R1
ADC R19, R4
RCALL DAA
MOV R1, R19

MOV R19, R0
ADC R19, R3
RCALL DAA
MOV R0, R19

//SIMPAN DI REGISTER TARGET
CLR R23
ADC R23, R23
MOV R22, R0
MOV R21, R1
MOV R20, R2

AKHIR: RJMP AKHIR

.ORG $80
VAR1: .DD $438291

.ORG $83
VAR2: .DD $999349
```

Hasil:

* 438291 + 999349 = 1437640

```
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x43
R20= 0x40  R21= 0x76
R22= 0x43  R23= 0x01
R24= 0x00  R25= 0x00
```

* 999999 + 999999 = 1999998

```
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x99
R20= 0x98  R21= 0x99
R22= 0x99  R23= 0x01
R24= 0x00  R25= 0x00
```

6. Program:

```
.include "m8535def.inc"
.org 0
RJMP MAIN
;Pemrogram : Amir Husein
;-------------------------
MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

//GET DATA DAT1
LDI ZL, LOW(2*DAT1)
LDI ZH, HIGH(2*DAT1)
LPM R3, Z+
LPM R2, Z+
LPM R1, Z+
LPM R0, Z+

//GET DATA DAT2
LDI ZL, LOW(2*DAT2)
LDI ZH, HIGH(2*DAT2)
LPM R7, Z+
LPM R6, Z+
LPM R5, Z+
LPM R4, Z+

//PENJUMLAHAN
MOV R19, R3
ADD R19, R7
RCALL DAA
MOV R3, R19

MOV R19, R2
ADC R19, R6
RCALL DAA
MOV R2, R19

MOV R19, R1
ADC R19, R5
RCALL DAA
MOV R1, R19

MOV R19, R0
ADC R19, R4
RCALL DAA
MOV R0, R19
//SIMPAN DI REGISTER TARGET
CLR R24
ADC R24, R24
MOV R23, R0
MOV R22, R1
MOV R21, R2
MOV R20, R3

AKHIR: RJMP AKHIR

.ORG $80
DAT1: .DD $43829199

.ORG $83
DAT2: .DD $99934999
```

Hasil:

* 43829199 + 99934999 = 143764198

```
R18= 0x00  R19= 0x43
R20= 0x98  R21= 0x41
R22= 0x76  R23= 0x43
R24= 0x01  R25= 0x00
```

* 99999999 + 99999999 = 199999998

```
R18= 0x00  R19= 0x99
R20= 0x98  R21= 0x99
R22= 0x99  R23= 0x99
R24= 0x01  R25= 0x00
```

7. Program:

```
.include "m8535def.inc"
.org 0
RJMP MAIN
;Pemrogram : Amir Husein
;-------------------------
MAIN:
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, HIGH(RAMEND)
OUT SPH, R16

//GET DATA DAT1
LDI ZL, LOW(2*DAT1)
LDI ZH, HIGH(2*DAT1)
LPM R4, Z+
LPM R3, Z+
LPM R2, Z+
LPM R1, Z+
LPM R0, Z+

//GET DATA DAT2
LDI ZL, LOW(2*DAT2)
LDI ZH, HIGH(2*DAT2)
LPM R9, Z+
LPM R8, Z+
LPM R7, Z+
LPM R6, Z+
LPM R5, Z+

//PENJUMLAHAN
MOV R19, R4
ADD R19, R9
RCALL DAA
MOV R4, R19

MOV R19, R3
ADC R19, R8
RCALL DAA
MOV R3, R19

MOV R19, R2
ADC R19, R7
RCALL DAA
MOV R2, R19

MOV R19, R1
ADC R19, R6
RCALL DAA
MOV R1, R19

MOV R19, R0
ADC R19, R5
RCALL DAA
MOV R0, R19
//SIMPAN DI REGISTER TARGET
CLR R25
ADC R25, R25
MOV R24, R0
MOV R23, R1
MOV R22, R2
MOV R21, R3
MOV R20, R4

AKHIR: RJMP AKHIR

.ORG $80
DAT1: .DQ $4382919992

.ORG $88
DAT2: .DQ $9993499992
```

Hasil:

* 4382919992+9993499992=14376419984

```
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x43
R20= 0x84  R21= 0x99
R22= 0x41  R23= 0x76
R24= 0x43  R25= 0x01
```

* 9999999999+9999999999=19999999998

```
R16= 0x02  R17= 0x00
R18= 0x00  R19= 0x99
R20= 0x98  R21= 0x99
R22= 0x99  R23= 0x99
R24= 0x99  R25= 0x01
```

V. ANALISIS

1. Pada program ini, data pada R22 perlu dipindahkan dahulu ke register R19, dikarenakan subrutin DAA hanya bisa memproses data yang ada pada register R19. Setelah data dipindahkan, register R19 yang sudah berisi data 2 digit dijumlahkan dengan data 2 digit pada register R23. Hasil yang didapat perlu dicek dan dilakukan faktor pengkoreksian dengan subrutin DAA, maka subrutin tersebut perlu dipanggil dengan

perintah RCALL. Setelah itu, hasil penjumlahan pada R19 yang tela dikoreksi dipindahkan ke R20, diikuti dengan instruksi ADC pada R21 untuk menambahkan carry (jika ada). Pada R21 ini tidak perlu dilakukan pengkoreksian dengan DAA dikarenakan angka maksimal yang akan muncul adalah 1 yang merupakan hasil penjumlahan maksimal dari 99 + 99 = 198.

2. Pada program nomor 2 ini berbeda dari sebelumnya, dikarenakan data yang dijumlahkan ada 10 terbentang dari R0 hingga R9. Untuk menyelesaikannya, diperlukan metode looping dengan pointer Z sebagai penunjuk ke alamat memori untuk R0 hingga R9. Untuk mengambil isi dari register R0 hingga R9 digunakan instruksi LD yang dimana akan disimpan di R28 untuk melakukan penjumlahan sementara. Program akan berhenti ketika nilai R30 sudah menyentuh $A, karena data yang dijumlahkan berjumlah 10 data.
3. Pada program ini, terdapat 2 data 4 digit R21:R20 dan R23:R22. R20 perlu dijumlahkan dengan R22, tapi operasi penjumlahan ini perlu dilakukan di R19, kemudian diberi faktor koreksi dengan memanggil subrutin DAA. Setelah itu hasilnya perlu dipindahkan ke R24. Langkah selanjutnya adalah menjumlahkan isi dari R21 dan R23 dengan instruksi ADC, serta tetap operasi ini dilakukan di R19, setelah itu hasilnya dipindahkan ke R25. R26 perlu diberi perintah ADC untuk menyimpan nilai carry yang ada.
4. Program pada soal ini sama seperti pada nomor 2, yang menjadi perbedaan hanya ada pada jumlah digit datanya dan disimpan di program area, bukan data area, sehingga dibutuhkan 2 register tambahan untuk menampung nilai data yang diambil dari program area. Register R27 bertindak sebagai counter dengan nilai maksimal 9, karena ketika nilai R27 telah mencapai 9, proses penjumlahan telah selesai. Data pada program area disimpan pada alamat awal $80.
5. Pada program ini, VAR1 dan VAR2 perlu dideklarasi dan diinisialisasi nilainya dahulu, bisa diawal program maupun diakhir program. Proses ini menggunakan directives .ORG untuk menentukan alamat pada program area, dengan tipe data yang digunakan adalah DD karena data yang dideklarasi merupakan data 6 digit. Langkah selanjutnya adalah melakukan pointing dengan register pointer Z serta melakukan pengisian atau pemindahan nilai pada program area ke GPR. Data 6 digit VAR1 akan disimpan di R0:R1:R2 dan data VAR2 disimpan di R3:R4:R5. Proses penjumlahan hanyalah menjumlahkan tiap register sesuai posisinya seperti halnya melakukan penjumlahan biasa, setelah itu diikuti oleh DAA untuk melakukan pengkoreksian. Perlu diingat bahwa setiap proses penjumlahan perlu dilakukan di R19, sehingga proses pemindahan nilai dari register ke register seringkali terjadi.
6. Program ini sama seperti program pada soal sebelumnya, yang menjadi perbedaan adalah jumlah digit masing-masing data yang akan dijumlahkan. Apabila pada soal sebelumnya 6 digit, maka pada soal ini merupakan 8 digit. Data pada DAT1 akan disimpan di register R0:R1:R2:R3 dan DAT2 di R4:R5:R6:R7, proses pengambilan data ini dilakukan dengan instruksi LPM pada pointer Z yang sudah tertuju pada alamat memori dari DAT1 dan DAT2. Proses penjumlahan hanyalah menjumlahkan tiap register sesuai posisinya seperti halnya melakukan penjumlahan biasa, setelah itu diikuti oleh DAA untuk melakukan pengkoreksian. Perlu diingat bahwa setiap proses penjumlahan perlu dilakukan di R19, sehingga proses pemindahan nilai dari register ke register seringkali terjadi.

7. Program ini sama seperti program pada soal sebelumnya, yang menjadi perbedaan adalah jumlah digit masing-masing data yang akan dijumlahkan. Apabila pada soal sebelumnya 8 digit, maka pada soal ini merupakan 10 digit. Data pada DAT1 akan disimpan di register R0:R1:R2:R3:R4 dan DAT2 di R5:R6:R7:R8:R9, proses pengambilan data ini dilakukan dengan instruksi LPM pada pointer Z yang sudah tertuju pada alamat memori dari DAT1 dan DAT2. Proses penjumlahan hanyalah menjumlahkan tiap register sesuai posisinya seperti halnya melakukan penjumlahan biasa, setelah itu diikuti oleh DAA untuk melakukan pengkoreksian. Perlu diingat bahwa setiap proses penjumlahan perlu dilakukan di R19, sehingga proses pemindahan nilai dari register ke register seringkali terjadi.

VI. DAFTAR PUSTAKA

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