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**

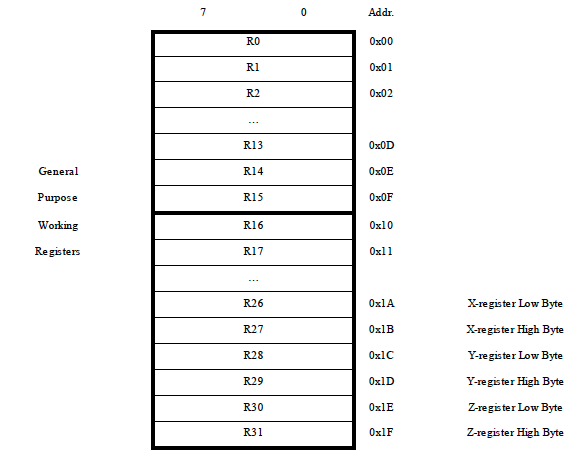
1. Mikrokontroler Atmega8535

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

1. 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.

1. 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 🡨 Rd –Rr;

Syntax: SUB Rd, Rr;

Operands: 0 <= Rd,Rr <= 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 🡨 Rd -Rr -C;

Syntax: SBC Rd, Rr;

Operands: 0 <= Rd,Rr <= 31

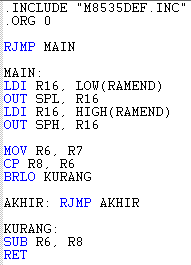
1. Instruksi LPM

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.

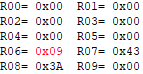
**III. SOAL LAB**

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 mengunakan 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.

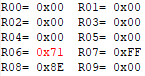
**IV. PROGRAM & HASIL**

1. Program: Hasil:

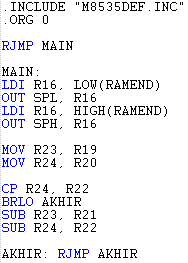
\* 43- 3A = 09



\* FF – 8E = 71



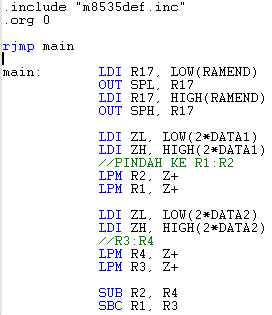
1. Program: Hasil:

 \* FFFF – AEAE = 5151

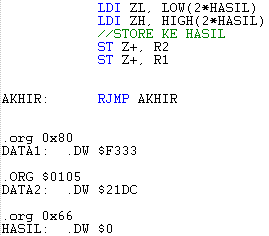


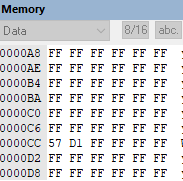
\* ABAB – 1212 = 9999



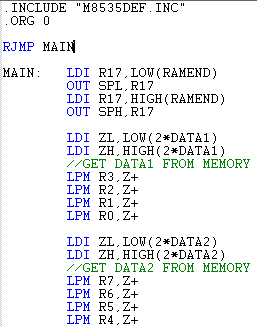
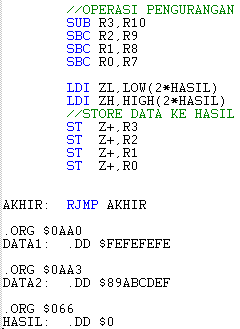
1. Program: Hasil:

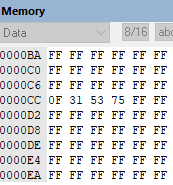
\* F333 – 21DC = D157

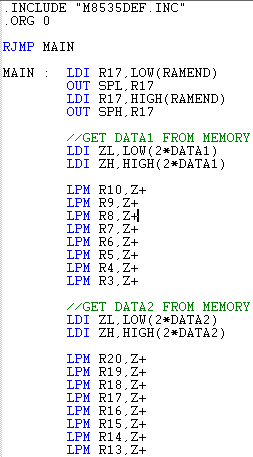
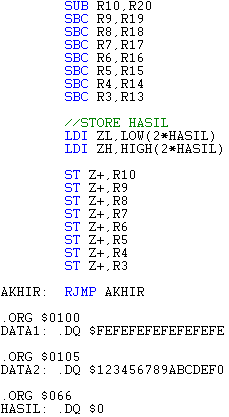
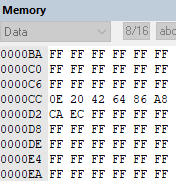




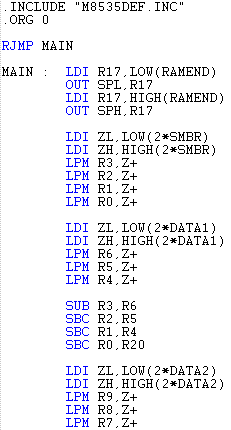
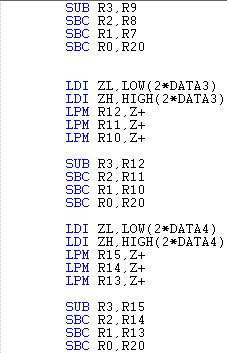
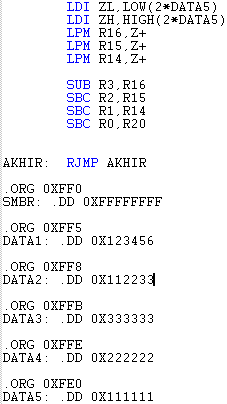
1. Program: Hasil:

 \* FEFEFEFE – 89ABCDEF = 7553310F



1. Program: Hasil:
2.  \*FEFEFEFEFEFEFEFE –
3. 123456789ABCDEF0 =
4. 

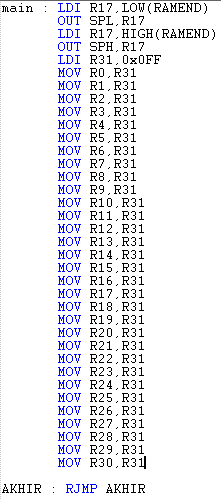
6. Program:

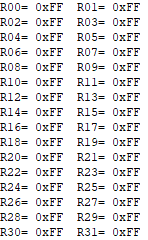


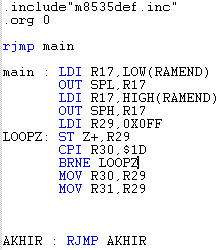
Hasil:

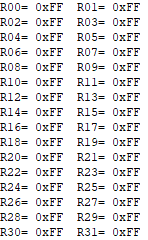
\* FFFFFFFF – 123456 – 112233 – 333333 – 222222 – 111111 = FF764310

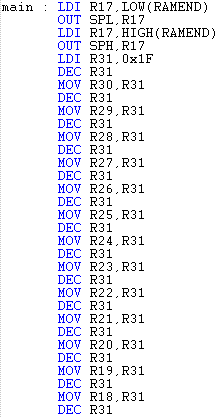
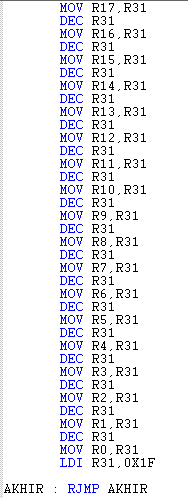


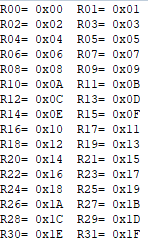
7A. Program: Hasil:



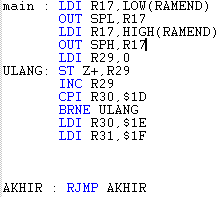
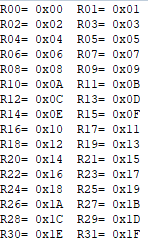
7B. Program: Hasil:



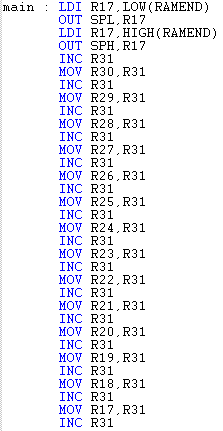
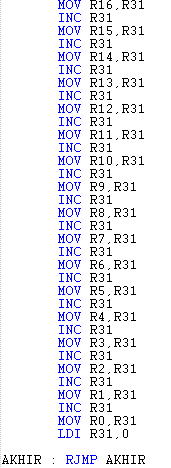
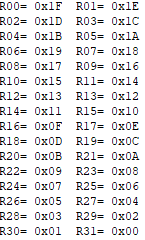
8A. Program: Hasil:



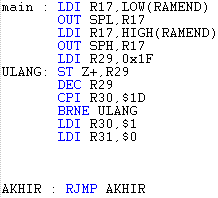
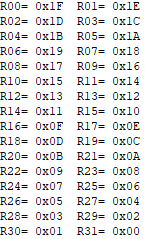
8B. Program: Hasil:



9A. Program: Hasil:

9B. Program: Hasil:

**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.

1. Program ini sebenarnya persis dengan program sebelumnya nomor 3. Letak perbedaanya 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.

1. Program ini pun sama persis seperti sebelumnya, lebar data yang digunakan kali ini untuk peroses 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.
2. 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 penguranganya. Setelah dipindah ke register, tinggal dilakukan pengurangan saja dengan SUB untuk 8-bit LSB dan sisanya menggunakan SBC.
3. 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.

1. 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.

1. A.) Pada program ini, dilakukan tanpa metoda loop, sehingga peru 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](https://yusrizalandeslubs.wordpress.com/dasar-elektronika/)). Diakses 5 Oktober 2020