PENERAPAN MODUS PENGALAMATAN DAN OPERASI ARITMATIKA PADA MIKROPROSESOR INTEL 8088/8086

[1]Ardi Panjaitan, [2]Masdiana Sagala
[1]Teknik Informatika Unika St. Thomas S.U; Jln. Setia Budi No.479-F Medan, 061-8210161
[2]Teknik Informatika Unika St. Thomas S.U; Jln. Setia Budi No.479-F Medan, 061-8210161
e-mail : Diana.sgl6 @gmail.com

Abstrak

Mikroprosessor merupakan salah satu bagian penting dalam mesin komputer dimana instruksi-instruksi dieksekusi. Dalam pembelajaran ilmu komputer, komunikasi dan cara kerja antar komponen dalam *mikroprosessor* merupakan bagian yang penting. Salah satu topik utamanya adalah pengalamatan register dan proses aritmatika yang terjadi dalam *mikroprosessor*, dan dapat diajarkan dengan bantuan simulasi. Jika ada suatu aplikasi yang dapat memberikan suatu simulasi mengenai pengalamatan *register* dan proses *aritmatika* dalam *mikroprosessor*, tentu akan lebih mudah untuk memahami cara kerja mikroprosesor.

Kata kunci: Mikroprosessor, Register, Aritmatika

Abstract

A microprocessor is an important part of a computer machine where instructions are executed. In learning computer science, communication and how to work between components in the microprocessor is an important part. One of the main topics is register addressing and arithmetic processes that occur in microprocessors, and can be taught with the help of simulations. If there is an application that can provide a simulation of addressing registers and arithmetic processes in a microprocessor, it would be easier to understand how the microprocessor works.

Keywords: Microprocessor, Register, Arithmetic

1. PENDAHULUAN

Pada saat era yang sudah maju seperti sekarang ini komputer sudah menjadi kebutuhan bagi banyak orang terutama bagi kalangan mahasiswa dan professional. Berbicara mengenai komputer tentu tidak akan lepas dari prosesor yang pada umumnya dikenal sebagai otaknya komputer. Jika komponen *PC* lainnya berfungsi sebagai pentransmisi data, maka prosesorlah yang berfungsi menentukan dan menghitung semua aktivitas tersebut.

Mikroprosesor Intel 8088 termasuk keluarga mikroprosesor 8 bit dan 16 bit. Mikroprosesor 8088 mempunyai 8 bit jalur data dan 20 bit jalur alamat. Jalur data memiliki pin yang sama dengan jalur alamat, artinya pada saat tertentu digunakan sebagai jalur data dan pada saat yang lain digunakan sebagai jalur alamat. 8088 ditargetkan pada sistem yang ekonomis, diikuti oleh pengunaan desain 8-bit. Jalur bus yang lebar dalam *sircuit boards* masih sangatlah mahal ketika ini di luncurkan. *Queue* yang ungul dari 8088 adalah 4 bytes, sebagai penggunaan dalam 8086 6 bytes. 8088 termasuk keturunan dari 80188, 80288, 80186, 80286, 80386, 80486, dan 80388, microcontroller seperti yang masih digunakan sekarang.

Prosesor memiliki bus alamat sebanyak 20 bit, yang berarti ia mampu mengalamati hingga 1.048.575 lokasi memori. Secara heksadesimal, jumlah ini dinyatakan sebagai angka

00000 sampai dengan FFFFF. Ini adalah alamat-alamat fisik (*physical addresses*) dari mikroprosesor. Untuk 8088 dan 8086 yang bus alamatnya terdiri dari 20 bit, otomatis penulisan alamat fisiknya terdiri dari 5 digit heksadesimal. Sistem segmentasi pada IBM PC dilaksanakan unik. 1 segment adalah bagian dari ruang memori yang besarnya 65536 byte atau 64 Kb. Namun, segment-segment itu tidaklah diletakkan secara berdampingan sambung menyambung satu sama lain, akan tetapi saling tumpang tindih sehingga jarak antara titik awal suatu segment hanya terpaut 16 byte terhadap segment lainnya.

Register dapat dibagi dalam lima golongan yaitu *general purpose register* (AX, BX, CX dan DX), *segment register* (CS, DS, SS dan ES), *pointer register* (IP, SP dan BP), *index register* (SI dan DI) dan *flag register*.

2. METODOLOGI PENELITIAN

II.1. Pembuatan Op-Code Secara Manual

1. Pembuatan Op-code atau susunan instruksi yang akan dilaksanakan prosesor meliputi: Tentukan perintah program yang akan dibuat kode operasinya. Contoh : **MOV BX, DX**
2. Lihat pada tabel instruction set (data sheet Intel 8088/86) perintah program yang kita buat. Dari banyak pilihan tambahan pada instruction set itu, ambil salah satu yang sesuai dengan perintah program. Dari contoh program diatas, dapat diketahui bahwa program diatas adalah program yang meng-copy data dari register DX ke register BX, sehingga dapat kita pilih **Register/Memory to/from Register**.

MOV - Move

Immediate to Reg	1011 wreg	data-LSB	data-MSB (w=1)	T=4
Register Direct	1000 10dw	modregr/m		T=2
Register Indirect	1000 10dw	modregr/m		T=13
Memory to Accumulator	1010 000w	addr-low	addr-high	T=14
Accumulator to Memory	1010 001w	addr-low	addr-high	T=14
Index	1000 10dw	modregr/m	disp	T=16

PUSH - Push

Register	0101 0reg	T=15
Segment Register	000reg110	T=14

POP - Pop

Register	0101 1reg	T=12
Segment Register	000reg111	T=12

XOR - Exclusive Or

Reg and Reg	0011 00dw	modregr/m			T=3
Immediate to Reg	1000 000w	mod110r/m	data-LSB	data (if w=1)	T=4
Immediate to AX/AL	0011 010w	data-LSB	data (w=1)		T=4

ADD - Add

Reg with Reg	0000 00dw	modregr/m			T=3
Immediate to Reg	1000 000w	mod000r/m	data-LSB	data(if w=1)	T=4
Immediate to AX/AL	0000 010w	data-LSB	data (w=1)		T=4

JMP - Unconditional Jump

Direct w/in Segment Short	1110 1011	disp	T=15
JZ - Jump on Zero	0111 0100	disp	T=16/4
JC - Jump on Carry	0111 0010	disp	T=16/4

```
d-direction: if d=1 then 'to' reg (Reg ← Mem)
             if d=0 then 'from' reg (Reg ← Reg, Mem ← Reg)
w-word: if w=1 then word operation (1 word = 2 bytes)
        if w=0 then byte operation
mod-mode:
      if mod=00 then DISP=0, disp-low and disp-high are absent
      if mod=01 then DISP=disp-low (signed), disp-high are absent
      if mod=10 then DISP=disp-low and disp-high
      if mod=11 then field r/m is translated as a register (REG field)
disp-displacement:
      show how far should the CPU jump from recent point (reg. IP)
r/m: if r/m = 111 then EA = (BX) + DISP
```

REG is assigned according to the following table:

16-Bit (w=1)	8-Bit (w=0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

(Syahrul, 2010, hal: 76)

3. Ambil kode instruksinya dan tuliskan. Contoh : kode instruksi **Register/Memory to/from Register** adalah **100010dw mod reg r/m**.

4. Gantilah variabel yang tidak diketahui seperti reg, w, data, dan lain-lain dengan digit biner yang sesuai. Variabel–variabel ini dapat dicari pada NOTES dari tabel instruction set, yaitu :
   - **a)** **d (direction) :**Menyatakan arah perpindahan data, d=1 bila data berpindah ke suatu register, d=0 bila data berpindah dari suatu register. Pada perintah diatas, d dapat berisi 1 atau 0, sehingga dapat dipilih salah satu. Misal kita pilih d=1, berarti data dinyatakan berpindah ke suatu register yaitu register BX.
   - **b)** **w (word) :** Menyatakan apakah operasi melibatkan data word (16 bit) atau tidak. Bila ya, maka w=1, bila tidak w=0. Perintah MOV BX,DX melibatkan word, karena baik BX maupun DX merupakan register 16 bit. Sehingga kita tentukan w=1.
   - **c)** **mod (mode) :** Merupakan mode lintas data, ada empat pilihan yang terdapat kata DISP melibatkanmemori, sehingga kita pilih mod=11 yang menyatakan operasi antar register.
   - **d)** **reg(register)** : Menyatakan jenis register yang digunakan. Karena kita telah menentukan d=1, yang berarti data berpindah ke register BX, maka kita cari kode biner untuk register BX, yaitu 011.
   - **e)** **r/m (register/memory**) :Menyatakan cara perhitungan physical address dari memori. Karena perintah program kita tidak melibatkan memori, maka kita boleh sembarang memilih. Misalkan kita pilih r/m=111. Dari keterangan NOTES diatas, kita dapatkan: **10001011 11011111.**
5. Konversikan kode biner yang telah diperoleh ke dalam bentuk heksadesimal. **1000 1011 1101 1111 = 8BDF**
6. Kode operasi telah didapat, yaitu : **8BDF**

II.2. Proses Eksekusi Instruksi

Instruksi - instruksi mikroprosesor INTEL 8088/8086 yang didukung di dalam perangkat lunak ini adalah modus pengalamatan (MOV), penjumlahan (ADD), penjumlahan dengan satu (INC), penjumlahan dengan *carry* (ADC), pengurangan (SUB), pengurangan dengan satu (DEC), pengurangan dengan pinjaman (SBB), perkalian (MUL) dan pembagian (DIV). Sedangkan *register – register* yang didukung adalah *general purpose register*, yaitu AX, BX, CX, DX, dan *register flag*, yaitu Z (*Zero*), C (*Carry*), A (*Half-Carry*), S (*Sign*), P (*Parity*) dan O (*Overflow*).

Sebagai contoh, ekspresi aritmatika = FFFFh – ((30h + 43h + 1A0h) * 2h) dapat dituliskan dalam bentuk *op-code* sebagai berikut :

1. MOV AX, 0030
2. MOV BX, 0043
3. MOV CX, 01A0
4. ADD AX, BX
5. ADD AX, CX
6. MOV BX, 0002
7. MUL BX
8. MOV BX, AX
9. MOV AX, FFFF
10. SUB AX, BX

Eksekusi dari setiap baris *op-code* beserta *register flag* yang berubah adalah sebagai berikut :

1. MOV AX, 0030

Instruksi MOV akan menyalin bilangan 0030H ke *register* AX.

Isi *register* AX sekarang = 0030H = 0000000000110000.
2. MOV BX, 0043
Instruksi MOV akan menyalin bilangan 0043H ke *register* BX.
Isi *register* BX sekarang = 0043H = 0000000001000011.
3. MOV CX, 01A0
Instruksi MOV akan menyalin bilangan 01A0H ke *register* CX.
Isi *register* CX sekarang = 01A0H = 0000000110100000.
4. ADD AX, BX
Instruksi ADD menjumlahkan isi dari *register* AX dan BX. Hasil penjumlahan dikembalikan ke register AX.
Isi dari *register* AX ditambah dengan isi *register* BX = 0000000000110000 + 0000000001000011 = 0000000001110011

```
        0000000000110000
        0000000001000011
    ---------------------------------- +
        0000000001110011
```

Isi dari *register* AX berubah menjadi 0000000001110011
Flag Z = 0 -> Hasil penjumlahan tidak nol.
Flag C = 0 -> bit *carry* dari operasi.
Flag A = 0 -> bit *half carry* dari operasi.
Flag S = 0 -> Hasil penjumlahan positif.
Flag P = 0 -> Jumlah Bit bernilai '1' = 5 (ganjil).
Flag O = 0 -> Hasil operasi tidak *overflow*.

5. ADD AX, CX
Instruksi ADD menjumlahkan isi dari *register* AX dan CX. Hasil penjumlahan dikembalikan ke *register* AX.
Isi dari *register* AX ditambah dengan isi *register* CX = 0000000001110011 + 0000000110100000 = 0000001000010011

```
        0000000001110011
        0000000110100000
      ---------------------------- +
        0000001000010011
```

Isi dari *register* AX berubah menjadi 0000001000010011
Flag Z = 0 -> Hasil penjumlahan tidak nol.
Flag C = 0 -> bit *carry* dari operasi.
Flag A = 1 -> bit *half carry* dari operasi.
Flag S = 0 -> Hasil penjumlahan positif.
Flag P = 1 -> Jumlah Bit bernilai '1' = 4 (genap).
Flag O = 0 -> Hasil operasi tidak *overflow*.

6. MOV BX, 0002
Instruksi MOV akan menyalin bilangan 0002H ke *register* BX.
Isi *register* BX sekarang = 0002H = 0000000000000010.

7. MUL BX
Instruksi MUL melakukan operasi perkalian isi dari *register* BX dengan AX. Hasil perkalian disimpan ke *register* DX-AX.
Isi dari register BX dikali isi *register* AX = 0000000000000010 * 0000001000010011 = 00000000000000000000010000100110

```
                                0000000000000010
                                0000001000010011
                              -------------------------------- *
                                0000000000000010
                               0000000000000010
                              0000000000000010
                             0000000000000010
                            ------------------------------------------ +
                             00000000000000010000100110
```

Isi dari *register* DX berubah menjadi 0000000000000000.
Isi dari *register* AX berubah menjadi 0000010000100110.
Flag Z = 0 -> Hasil perkalian tidak nol.
Flag C = 0 -> MSB 16 bit = 0
Flag S = 0 -> Hasil perkalian positif.
Flag P = 1 -> Jumlah Bit bernilai '1' = 4 (genap).
Flag O = 0 -> Hasil operasi tidak *overflow*.

8. MOV BX, AX
Instruksi MOV memindahkan data dari *register* AX ke BX.
Isi *register* BX sekarang = isi dari *register* AX = 0000010000100110 = 0426H.

9. MOV AX, FFFF
Instruksi MOV akan menyalin bilangan FFFFH ke *register* AX.
Isi *register* AX sekarang = FFFFH = 1111111111111111.

10. SUB AX, BX
Instruksi SUB mengurangkan isi dari *register* AX dan BX. Hasil pengurangan dikembalikan ke register AX.
Isi dari *register* AX dikurang dengan isi *register* BX = 1111111111111111 - 0000010000100110 = 1111101111011001

```
                    1111111111111111
                    0000010000100110
                ---------------------------------  -
                    1111101111011001
```

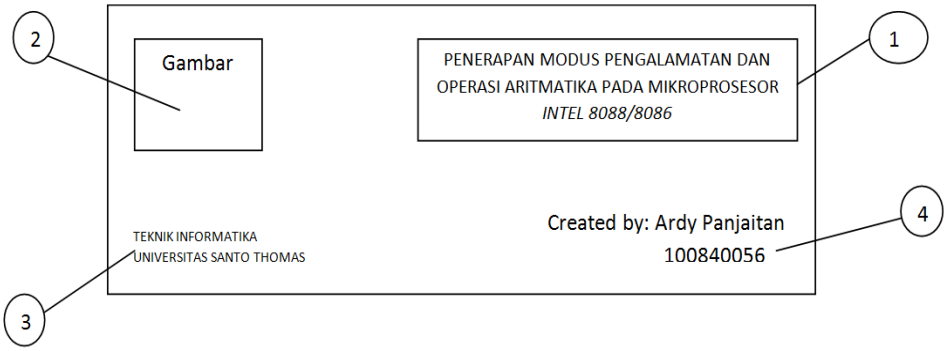
Isi dari *register* AX berubah menjadi 1111101111011001
Flag Z = 0 -> Hasil pengurangan tidak nol.
Flag C = 0 -> bit *carry* dari operasi.
Flag A = 0 -> bit *half carry* dari operasi.
Flag S = 0 -> Hasil pengurangan positif.
Flag P = 1 -> Jumlah Bit bernilai '1' = 12 (genap).
Flag O = 0 -> Hasil operasi tidak overflow.

Hasil operasi disimpan pada *register* AX, yaitu 1111101111011001 atau FBD9(heksa) atau 64473 (desimal).

3. HASIL DAN PEMBAHASAN

III.1. Hasil

Pada Gambar 1 berfungsi sebagai *form* awal (pembuka) pada perangkat lunak. Klik pada *form* atau ketik tombol '<ENTER>' untuk melanjutkan

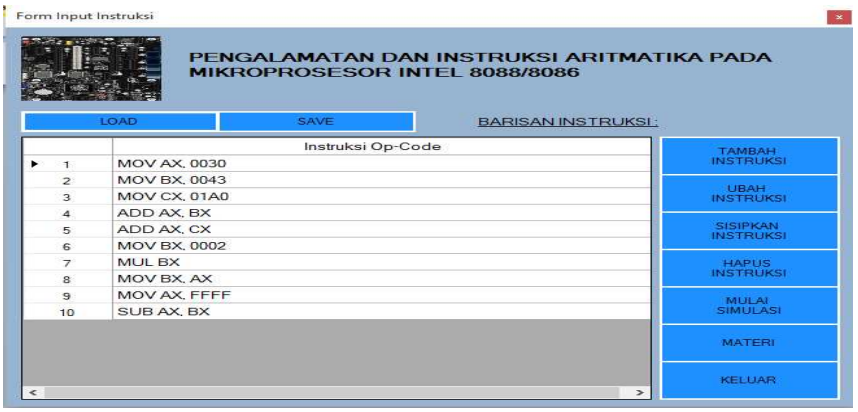


Gambar 1. Form Splash Screen

III.2 Pembahasan

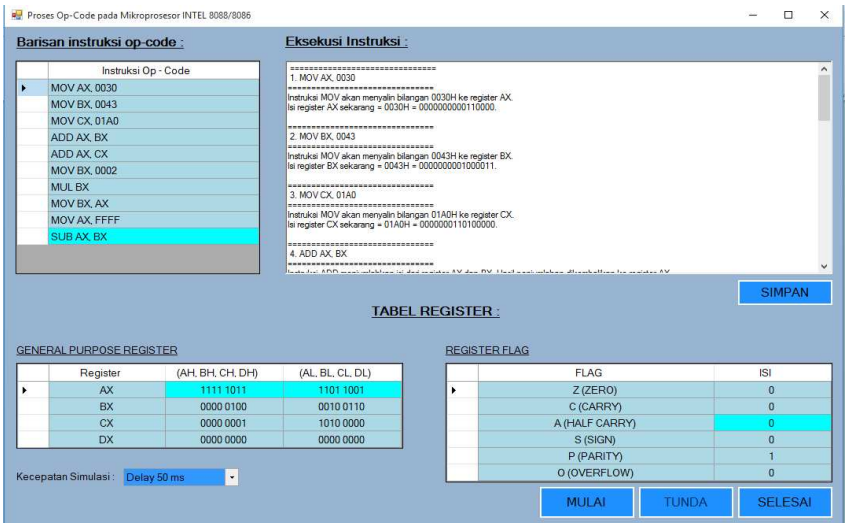
Form input berfungsi untuk menampung barisan instruksi, dengan menekan tombol load maka akan muncul barisan instruksi yang sebelumnya sudah disimpan dalam format "Mic.". Kemudian tombol berikutnya yang ada pada form input berfungsi sebagai berikut:

1. Tambah Instruksi : Merupakan tombol untuk menampilkan form tambah instruksi.
2. Ubah Instruksi : Merupakan tombol untuk menampilkan form ubahinstruksi
3. Sisipkan Instruksi : Merupakan tombol untuk menampilkan form sisipkaninstruksi.
4. Hapus Instruksi : Merupakan tombol untuk menampilkan form hapusinstruksi.
5. Mulai Simulasi : Merupakan tombol untuk menampilkan form prosesInstruksi.
6. Materi : Merupakan tombol untuk menampilkanform materi.
7. Save : Merupakan tombol untuk menyimpanbarisan instruksiyang sudah berhasil di eksekusi.Form input dapat dilihat pada gambar 2 berikut



Gambar 2. Input Instruksi

Form proses merupakan form yang akan mengeksekusi semua barisan instruksi yang sudah di input. Pada form proses akan ditampilkan keterangan dan hasil eksekusi baris demi baris. Form proses dapat dilihat pada gambar 3 berikut.



Gambar 3 Form Proses

4. KESIMPULAN

Berdasarkan hasil-hasil analisis yang dilakukan maka kesimpulan yang dapat diambil adalah sebagai berikut :

1. Perangkat lunak ini dapat digunakan untuk memahami modus pengalamatan *register* dan segera serta beberapa operasi aritmatika pada mikroprosesor intel 8088/8086.
2. Perangkat lunak dapat dijadikan sebagai media pendukung dalam proses pembelajaran mikroprosesor dasar.

5. SARAN

1. Modus pengalamatan yang dibahas dapat diperbanyak seperti menambahkan modus pengalamatan langsung, tidak langsung, *base-plus-index*, *register* relatif dan indeks berskala.
2. Operasi yang dibahas dalam perangkat lunak dapat diperbanyak seperti menambahkan operasi aritmatika lainnya, operasi perbandingan, dan logika aritmatika.

DAFTAR PUSTAKA

1) Albert Paul Malvino, 1994, Elektronika Komputer Digital : *Pengantar Mikrokomputer, Edisi Kedua*, Penerbit Erlangga.

2) Barry B. Brey, 2003, Mikroprosesor Intel 8086/8088/80186/80188/80286/ 80386/80486 : *Arsitektur Pemrograman Antarmuka, Edisi 5, Jilid 2*, Penerbit Erlangga.

3) Djoko Sugiono, 2013, *Teknik Mikroprosesor,* penerbit Kementrian Pendidikan Dan Kebudayaan.

4) Gunawan Putrodjojo, Eko Budi Purwanto, 2003, *Mengoptimalkan Sistem Penyimpanan (Memory Sistem) Untuk Meningkatkan Kinerja Computer,* Jurnal Ilmiah Ilmu Komputer, Vol 1.

5) Hadi, Rahadian, 2006, Pemrograman *Microsoft Visual Studio 2005 dengan menggunakan Windows API*, PT. Elex Media Komputindo, Jakarta.

6) Muchlas, 2005 , *Rangkaian Digital*, Penerbit Gava Media, Yokyakarta.

7) Syahrul, 2010 , *Organisasi dan Arsitektur Komputer,* Penerbit Andi, Yokyakarta.