BUKU PANDUAN PRAKTIKUM

Buku Panduan Praktikum
Sistem Berbasis Mikroprosesor

Penulis:
Dr. Muchlas, M.T.
Nuryono Satya Widodo, M.Eng.
Dr. Budi Santosa
Pramudita Budiastuti, M. Pd.

UAD
Universitas
Ahmad Dahlan

Yogyakarta 2020

Panduan Praktikum
Sistem Berbasis Mikroprosesor

Penulis :
Dr. Muchlas, M.T.
Nuryono Satya Widodo, S.T., M.Eng.
Dr. Budi Santosa
Pramudita Budiastuti, M. Pd.

Universitas Ahmad Dahlan
Yogyakarta
2020

Kata Pengantar

Alhamdulillah puji syukur dipanjatkan kepada Allah SWT yang telah memberikan petunjukNya sehingga penulisan "Buku Panduan Praktikum: Sistem Berbasis Mikroprosesor" yang dilakukan penulis pada masa-masa pandemi Covid-19 dapat diselesaikan dengan baik.

Secara umum buku ini terdiri dari 5 bab. Bab I membahas terkait petunjuk penggunaan: AxIDE & Programmer's Notepad. Pada bab II membahas terkait penggunaan simulator wookie 68HC11. Bab III menjelaskan terkait eksplorasi sistem 68HC11 dengan komputer host. Model pengalamatan sistem 68HC11 dibahas pada Bab IV. Bab V membahas terkait pemrograman port input/output pada EVBU M68HC11.

Kepada semua pihak yang telah membantu penyusunan panduan ini diucapkan terimakasih. Semoga bantuan tersebut menjadi amal sholeh dan mendapat imbalan pahala dari Allah Subhanahu wa ta'ala.

Dengan berbagai kekurangannya, panduan ini diharapkan dapat memberikan manfaat sesuai dengan fungsinya. Masukan-masukan dari siapapun sangat dinanti demi perbaikan panduan ini.

Yogyakarta, Juli 2020

Dr. Muchlas, M.T.
Nuryono Satya Widodo, S.T., M.Eng.
Dr. Budi Santosa
Pramudita Budiastuti, M. Pd.

Daftar Isi

BAB 1
PETUNJUK PENGGUNAAN

AxIDE & Programmer's Notepad

Pada dasarnya AxIDE adalah sebuah Program Terminal/komunikasi yang digunakan untuk melakukan komunikasi antara PC dengan EVBU M68HC11. Program ini serupa dengan Procomm maupun Hyperterminal, namun AxIDE dirancang untuk beroperasi pada lingkungan under-Windows dan memiliki menu khusus untuk penggunaan dengan EVBU. Cara menggunakan AxIDE, pada prinsipnya sama dengan Procomm.

A. Konfigurasi Awal AxIDE

1. Buka AxIDE dari shorcutnya atau dari Start Menu
2. Click tanda centang (check) yg terdapat di bagian kiri atas Window Program

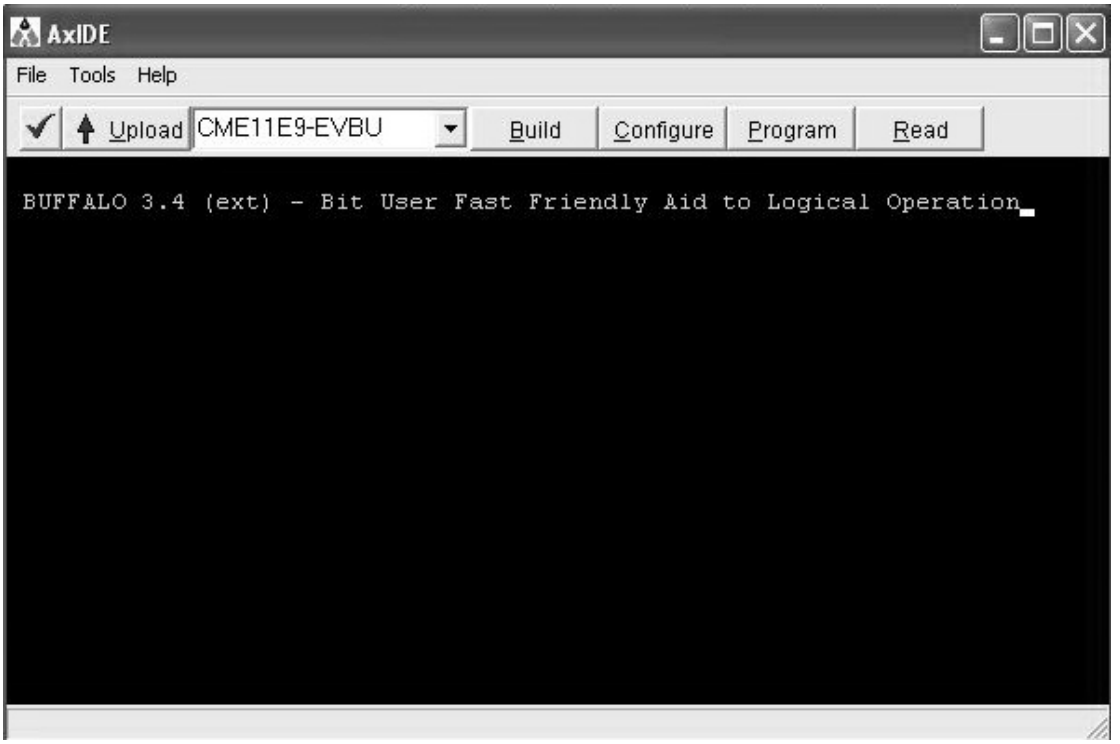
Gambar 1. Tampilan Konfigurasi Awal AxIDE.

3. Pastikan setting Port yang dipakai sesuai dengan Tabel 1 berikut ini:

Tabel 1. Setting Port Konfigurasi Awal AxIDE

Port	COM1/COM2	Handshaking	Information
Baud Rate	9600		
Parity	None	Xon/Xoff	OFF
Data Bits	8	Rts/Cts	OFF
Stop Bits	1	Dtr/Dsr	OFF

4. Pilih COM PORT yg sesuai dengan kabel serial yg terpasang pada PC
5. Jika telah sesuai, click OK
6. Pastikan yg dipilih pada drop down menu adalah CME11E9-EVBU
7. Pastikan AxIDE sudah dikonfigurasi dengan benar
8. Hubungkan kabel serial dari PC dengan EVBU
9. Hubungkan catu daya ke EVBU
10. Jika semua terkonfigurasi dengan baik dan tidak ada gangguan hardware pada PC maupun EVBU, maka akan tampak sebagai berikut:



Gambar 2. Tampilan Konfigurasi Awal AxIDE Bekerja Normal

11. Selanjutnya tekan Enter maka akan tampak Prompt sebagai berikut:

Gambar 3. Tampilan Akhir Konfigurasi Awal AxIDE.

Jika anda tidak memperoleh tampilan seperti ini periksa Kembali hardware anda dan konfigurasi pada AxIDE (Prosedur troubleshooting ada pada petunjuk troubleshooting).

B. Penggunaan AxIDE

1. Untuk melakukan assembly dan kompilasi anda ikuti langkah-langkah berikut:
    a. Click menu Build pada AxIDE, pilih file assembler (*.asm) yang akan anda kompilasi, gunakan tab browse untuk mengarahkan pada directory file anda. Click pada file yang anda maksud dan click Open (double click pada file tersebut).
    b. Lanjutkan dengan Click OK.
    c. Pastikan muncul window baru yang berisi file berekstensi *.lst dengan nama sama dengan file assembly anda.

d. Jika tidak ada kesalahan maka hasil kompilasi akan tampak pada window baru tersebut, jika terdapat kesalahan maka windowsyang berisi file berekstensi).lst tidak akan muncul namun suatu window baru akan menampilkan pesan kesalahan yang terjadi.

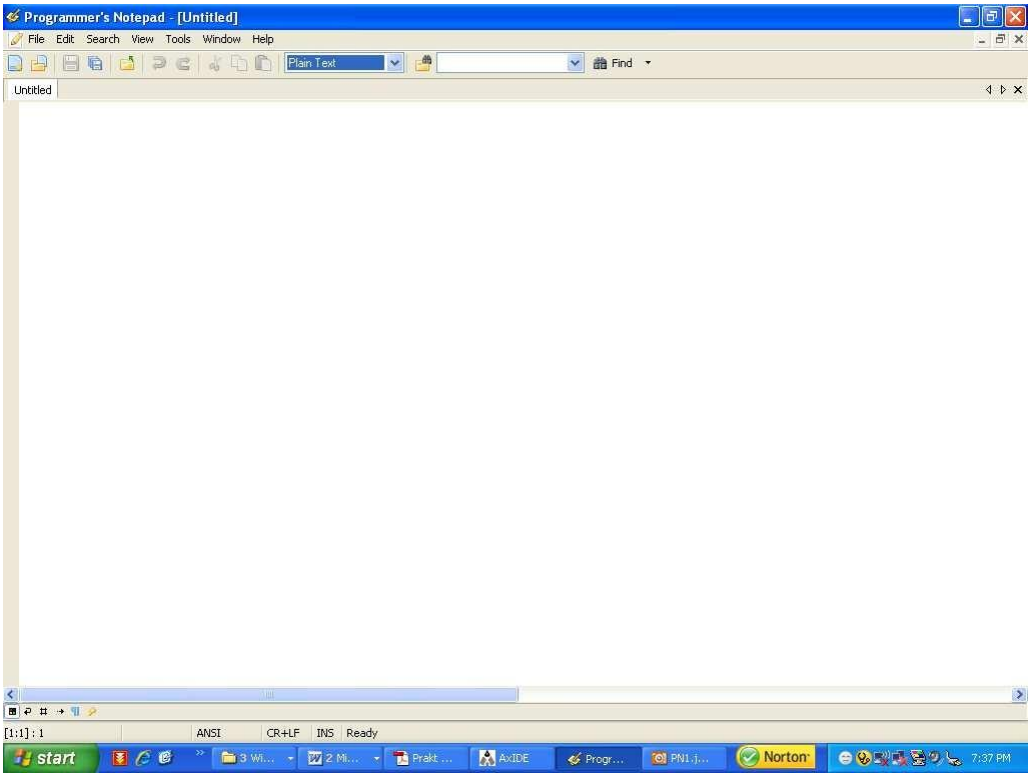
2. Untuk melakukan download ke EVBU, ikuti langkah berikut:
   a. Pastikan prompt BUFFALO sudah muncul pada AxIDE
   b. Ketikkan LOAD T, diikuti penekanan tombol Enter
   c. Click Upload pilih file S19 (*.s19) yang akan anda download, gunakan tab browse untuk mengarahkan pada directory file anda. Click pada file yang anda maksud dan click Open (double click pada file tersebut).
   d. Selanjutnya akan muncul informasi tentang proses download, baris yang telah berhasil di download ke dalam EVBU.
   e. Tunggu sampai muncul tulisan Done, dan prompt.
3. Untuk melakukan running program pada EVBU, setelah langkah 2 selesai, pada prompt yang muncul ketikkan G XXXX atau GO XXXX atau CALL XXXX, dengan XXXX adalah alamat awal program anda.

Aturan tentang directory
Untuk kemudahan pada praktikum anda akan diminta membuat suatu direktori (folder) tersendiri di drive D. Pastikan bahwa nama folder yang anda gunakan adalah HC11_xx dengan xx adalah nomor kelompok anda. AxIDE hanya bisa bekerja pada direktori/path yang tidak lebih dari delapan karakter.

C. Petunjuk Penggunaan Programmer's Notepad

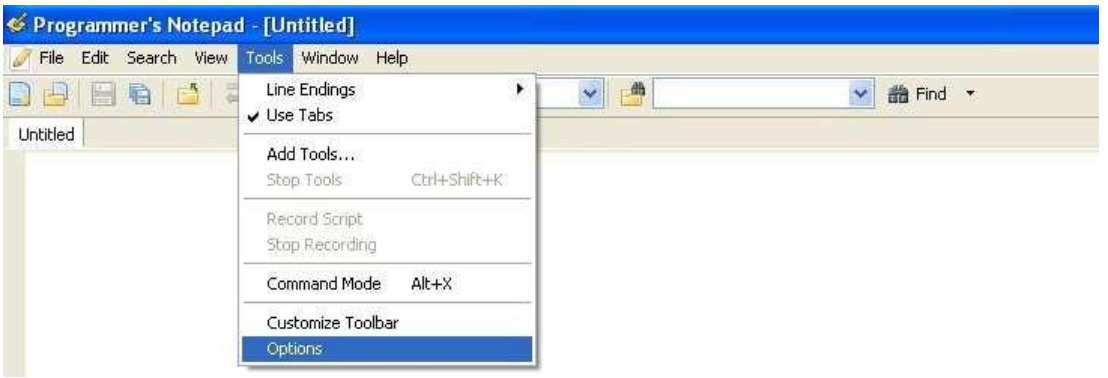
Sesuai dengan namanya software Programmer's Notepad adalah suatu software yang diperuntukkan bagi penulisan program dalam berbagai bahasa pemrograman. Pada praktikum dengan menggunakan EVBU M68HC11, kita akan menulis program dengan menggunakan bahasa assembly.

1. Cara menulis program.
   a. Buka software Programmer's Notepad, maka akan anda peroleh tampilan awal sebagai berikut:



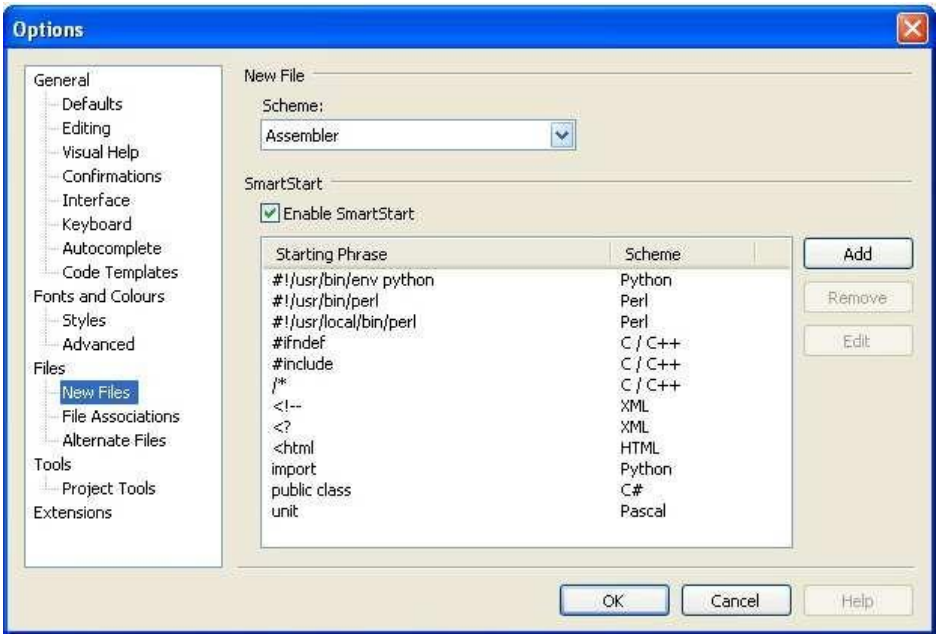
Gambar 4. Tampilan Awal Programmer's Notepad

b. Lakukan pengaturan pada Programmer's Notepad agar memudahkan penggunaan selanjutnya. Melalui menu **Tool>Option**



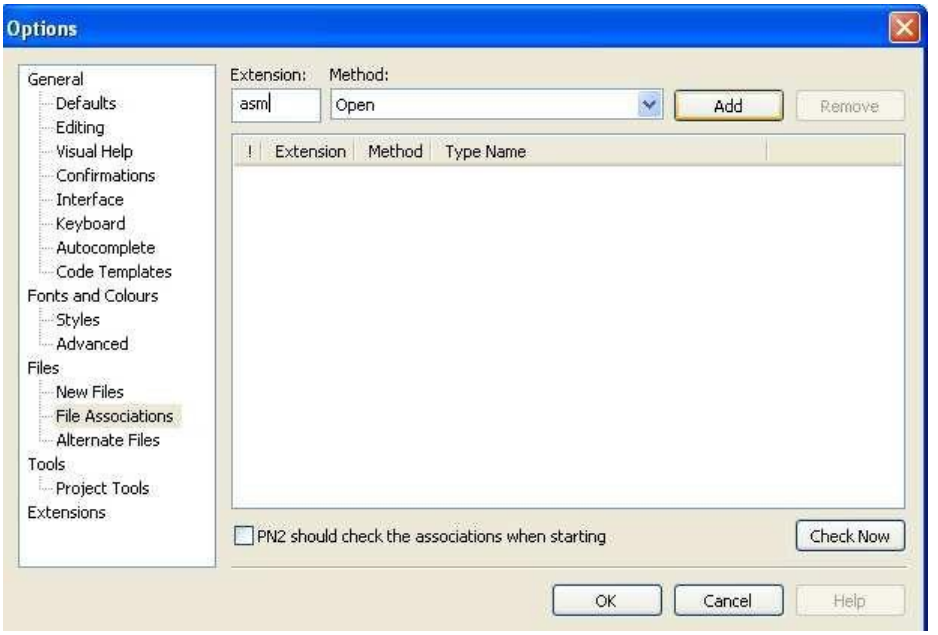
Gambar 5. Pengaturan Programmer's Notepad

c. Atur agar secara default Programmer's Notepad selalu menggunakan skema assembler dan format file *.asm (bahasa assembly), melalui tab New Files, pilih Assembler pada pull down menu **Scheme.** (seperti tampak pada gambar berikut)



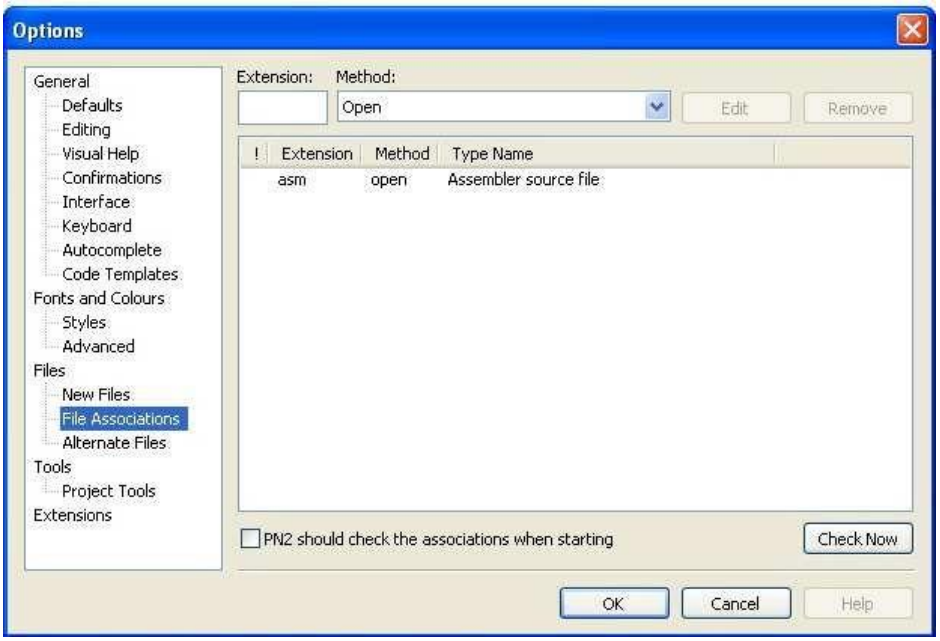
Gambar 6. Pengaturan Default Programmer's Notepad

Lanjutkan dengan mengetikkan asm dan click **add** pada tab **File associations** (seperti pada gambar berikut)



Gambar 7. Pengaturan Default Programmer's Notepad dengan mengetikkan asm dan click **add** pada tab **File associations**

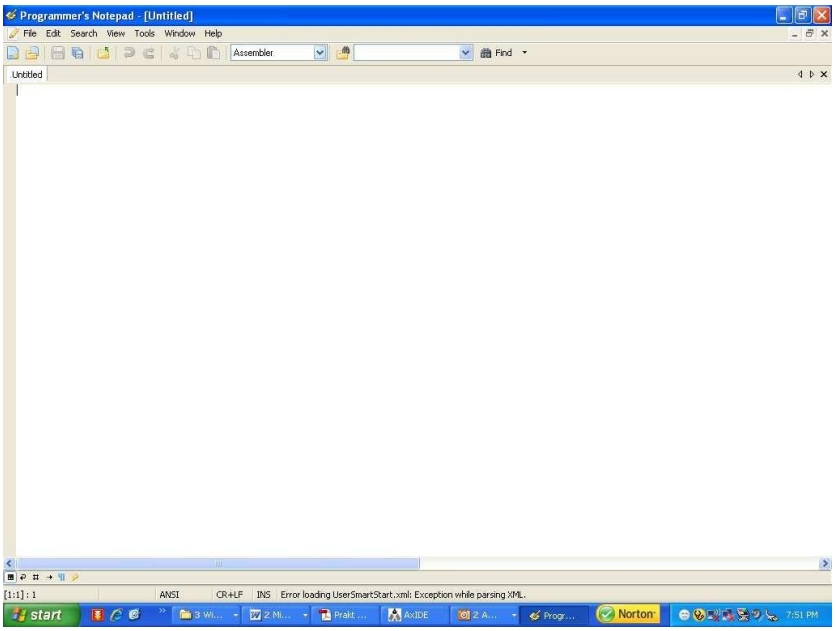
Maka akan muncul tampilan sebagai berikut:



Gambar 8. Tampilan Akhir Pengaturan Default Programmer's Notepad

Simpan pengaturan yang telah anda lakukan dengan click OK. Lanjutkan dengan menutup software Programmer's Notepad.

d. Buka kembali Programmer's Notepad, jika pengaturan yang anda lakukan telah benar maka akan muncul tampilan awal sebagai berikut



Gambar 9. Tampilan Programmer's Notepad Bekerja Normal

e. Programmer's Notepad telah siap untuk anda gunakan menulis program dalam bahasa assembly.
f. Untuk menulis Program dalam bahasa assembly, perhatikan aturan berikut. Suatu program dalam bahasa assembly sekurang-kurangnya memiliki tiga kolom, masing-masing kolom ditempati elemen program sebagaimana tampak pada contoh berikut ini.

```
Label & Identifier   Assembler Directives   Operand Komentar
Label & Identifier   Mnemonics              Operand Komentar
Label & Identifier   Mnemonics              Operand Komentar
Label & Identifier   Mnemonics              Operand Komentar
```

g. Untuk berpindah dari kolom Label&Identifier gunakan tombol tab, antara Assembler Directives atau Mnemonics dengan Operand anda dapat gunakan spasi atau tab.
h. Setelah selesai menulis program simpan dengan ekstensi(format) *.asm. Perhatikan pada pemberian nama file, nama file yang diijinkan adalah maximal 8 karakter.
i. Template program disediakan
j. Untuk selanjutnya anda hanya perlu mengikuti langkah 4 s/d selesai.

D. Buffalo Command

Daftar Instruksi pada Buffalo tersedia pada Tabel 2 berikut ini:

Tabel 2. Daftar Instruksi Pada Buffalo

Buffalo Command	Result
ASM [<address>]	Assembler/disassembler
BF <addr1> <addr2> <data>	Block fill memory with data
BR [-] [<address>]	Breakpoint set
BULK	Bulk Erase EEPROM
BULKALL	Bulk Erase EEPROM+CONFIG register
CALL [<address>]	Execute subroutine
G [<address>]	Execute program
HELP	Display monitor command
LOAD <host download command>	Download (S-Records*) via host port
LOAD T	Download (S-Records*) via terminal port
MD [addr1>[<addr2>]]	Dump memory to terminal
MM [<address>]	Memory modify
MOVE <addr1>< addr2>[,dest>]	Move memory to new location
P	Proceed/continue form breakpoint
RM [p, y, x, a, b, c, s,]	Register modify
T [<n>]	Trace $1-$FF instructions
TM	Enter transparent mode
VERIFY <host download command>	Compare memory to download via host port
VERIFY <T>	Compare memory to downliad via terminal

E. Troubleshooting

Gunakan Tabel 3 berikut ini untuk mengidentifikasi kesalahan atau kerusakan yang terjadi.

Tabel 3. Daftar Troubleshooting

No.	Gejala	Pemeriksaan/tindakan	Keterangan
1.	BUFFALO tidak muncul	Tekan tombol reset EVBU	
		Periksa Power Supply EVBU dan kabelnya	Ukur tegangan pada Test Point/ soket ADC (5V)
		Periksa Kabel serial	Cek pemasangan kabel pada COM PORT PC dan EVBU, ganti dengan kabel serial lain
		Cek setting komunikasi	Baudrate, protokol, dan COM yang dipakai
2.	Gagal Download s19 (rom-xxxx)	Cek mode operasi	Periksa jumper J3 dan J6
		Cek memory EVBU	Gunakan perintah Block Fill BF XXXX XXXX AA, cek bahwa setelah perintah ini alamat XXXX s/d XXXX berisi data AA (gunakan perintah MD XXXX)
3.	Gagal kompile (unrecognized Mnemonics)	Cek program anda	Terjadi kesalahan penulisan program
4.	Gagal kompile (can't open *.lst)	Cek nama folder penyimpan program anda atau nama file	AxIDE tidak dapat menemukan folder karena nama yg terlalu panjang
5.	Gagal kompile (as: can't open *.ASM)	Cek nama folder penyimpan program anda atau nama file	AxIDE tidak dapat membuka file karena nama yg terlalu panjang

PADA PRAKTIKUM MENGGUNAKAN EVBU di LABORATORIUM TEKNIK ELEKTRO SEMUA PROGRAM YANG ANDA BUAT AKAN DILETAKKAN di DALAM RAM, MAKA JANGAN MELAKUKAN DOWNLOAD DENGAN MEMILIH MENU PROGRAM YG ADA PADA TOOLBAR, MENU TERSEBUT DIGUNAKAN HANYA JIKA ANDA MEMPROGRAM EPROM ATAU EEPROM. SEDANGKAN MENU CONFIGURE HANYA DIPERGUNAKAN JIKA ANDA MENGOPERASIKAN EVBU PADA MODE SPECIAL (TEST & BOOTSTRAP) YANG TIDAK KITA PERLUKAN PADA PRAKT. SELALU GUNAKAN MENU UPLOAD UNTUK DOWNLOAD PROGRAM ke EVBU.

HARAP DIPERHATIKAN

BAB 2
PERCOBAAN I

Penggunaan Simulator Wookie 68HC11

A. Kompetensi Dasar

Mahasiswa menguasai pendahuluan bahasa assembly pada M68HC11

B. Indikator Pencapaian Kompetensi

1. Mengenal EVBU M68HC11, Wookie, Software Programmer's Notepad, AxIDE
2. Memanfaatkan beberapa fasilitas AxIDE
3. Menulis program Programmer's Notepad
4. Mengkompilasi instruksi assembly menjadi S-record dengan compiler AxIDE
5. Melakukan Simulasi dengan Wookie

C. Dasar Teori

Simulator Wookie (Wireless Object Oriented Kindly Interfaced Emulator) dibuat untuk membantu para programmer bahasa assembly mesin 68HC11 dari Motorola. Dengan menggunakan simulator Wookie, pemrogram dapat melakukan perbaikan program (debugging) maupun pengetesan program secara mudah dan benar. Oleh karena simulator Wookie ini dapat dijalankan tanpa perlu melibatkan perangkat keras dari mesin 68HC11 yang sesungguhnya, maka pemrogram dapat terhindar dari masalah-masalah yang berhubungan dengan penyediaan perangkat keras maupun masalah-masalah interkoneksinya. Dengan kata lain, simulator Wookie ini telah menyediakan secara virtual mesin 68HC11. Melalui simulator Wookie ini dapat ditampilkan semua resources mikrokontroler 68HC11 pada layar monitor komputer, sehingga pemrogram seolah-olah telah berhadapan langsung dengan mikrokontroler 68HC11 yang sesungguhnya.

Telah banyak tersedia simulator mikrokontroler 68HC11 seperti SIM68 yang dibuat oleh Perry J. Fisch. Sampai tahun 1995 SIM68 telah mencapai versi 2.21U, namun simulator tersebut pengoperasiannya tidak praktis. Wookie merupakan simulator 68HC11 yang mudah

pengoperasiannya dibuat oleh Kale Andersen dan kawan-kawan di bawah arahan Dr. Steven L. Barnicki dari Milwaukee School of Engineering dan sampai tahun 2002 pengembangannya telah mencapai versi 1.67. Seluruh resources yang meliputi simulator, compiler maupun downloader dari 68HC11 bersifat freeware dan dapat diperoleh secara gratis dengan cara download melalui situs Internet: http://www.msoe.edu/eecs/ce/ceb/ resources/.

D. Tugas Pendahuluan

1. Jelaskan pengertian mikrokontroler dan gambarkan diagram bloknya!
2. Kemukakan alasan-alasan mengapa mikrokontroler disebut sebagai on-chip- memory dan on-chip-perpheral!
3. Jelaskan tahap-tahap pemrograman mikrokontroler! Lengkapi penjelasan anda dengan flow chart!
4. Sebutkan nama dan spesifikasi register-register yang menjadi model pemrograman sistem 68HC11, serta jelaskan fungsi masing-masing register tersebut!
5. Sebutkan dan jelaskan jenis mode operasi mikrokontroler 68HC11!
6. Tulislah format penulisan instruksi assembly mikrokontroler 68HC11, dan berikan sebuah contoh instruksi agar 68HC11 melakukan penjumlahan 2 buah
7. Bilangan $2 dan $3 serta hasilnya disimpan pada alamat $0120! Anggap program dimulai pada alamat $0100 dan anggap pula instruksi-instruksi tersebut anda simpan ke dalam file COBA.ASM.

```
ORG     $0100
LDAA    #$2
ADDA    #$3
STAA    $0120
```

8. Perhatikan isi file COBA.LST hasil kompilasi berikut ini:

```
0001 0100                ORG          $0100
0002 0100 86 05                LDAA         #$05
0003 0102 8b 02                ADDA         #$02
0004 0104 b7 01 20             STAA         $0120
```

Isilah tabel peta memori 68HC11 berikut ini yang diperoleh berdasarkan listing program tersebut!

Tabel 4. Percobaan Penggunaan Simulator Wookie 68HC11

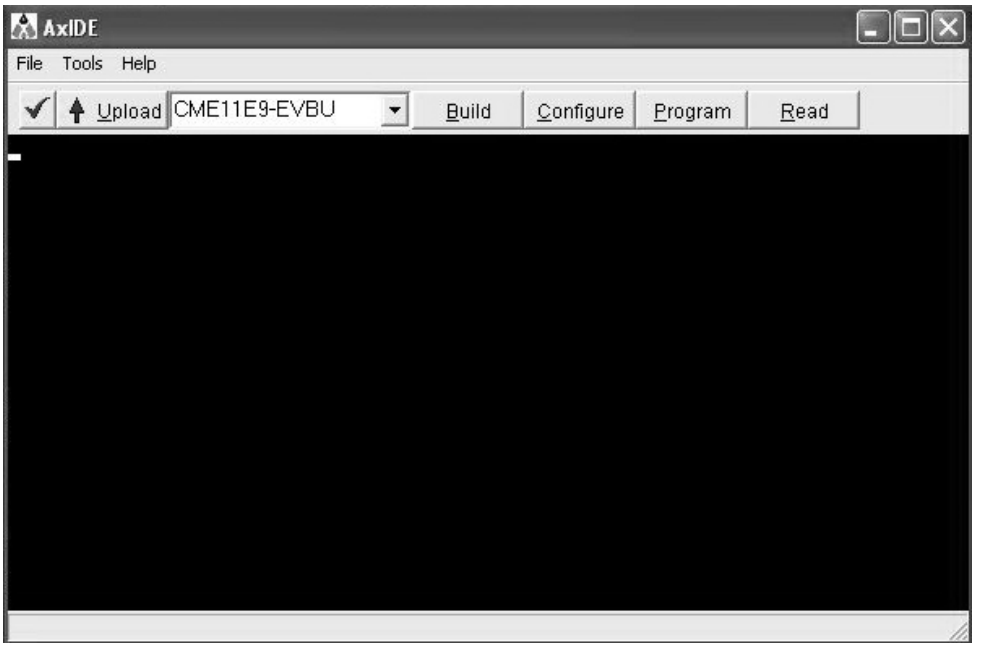
Alamat Memori	Instruksi dalam Heksadesimal	Instruksi Assembly	
		OPCODE	OPERAND
$0100	….	LDAA	
$0101	….		#$05
$0102	….	ADDA	
$0103	….		….
$0104	B7	….	
$0105	01		….
$0106	20		

E. Alat-alat dan Langkah Percobaan

1. Software AxIDE
2. Software Programmer's Notepad
3. Software/simulator Wookie

F. Persiapan Percobaan

1. Aktifkan/Run program AxIDE.



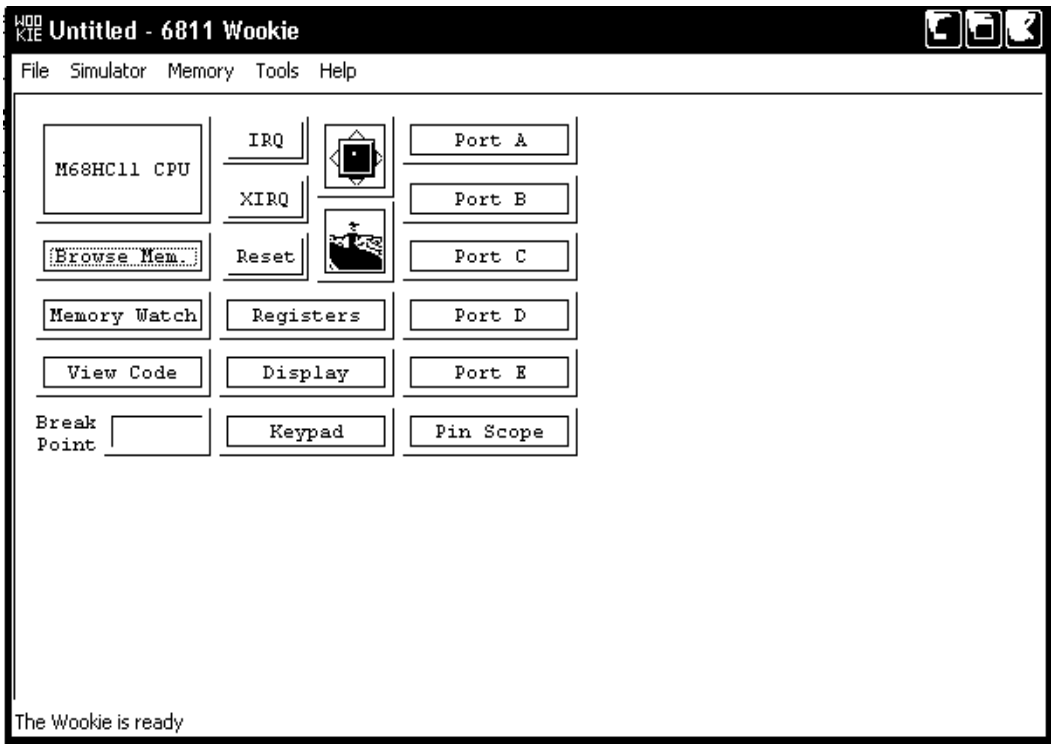
Gambar 10. Tampilan Program AxIDE

a. Menggunakan software Programmer's Notepad, tulis program berikut ini:

```
ORG     $0100
LDAA    #$2
ADDA    #$3
STAA    $0120
```

Simpan dengan nama COBA.asm. Lakukan assembly melalui program AxIDE. (Ikuti petunjuk halaman 3 tentang assembly dan kompilasi dengan AxIDE)

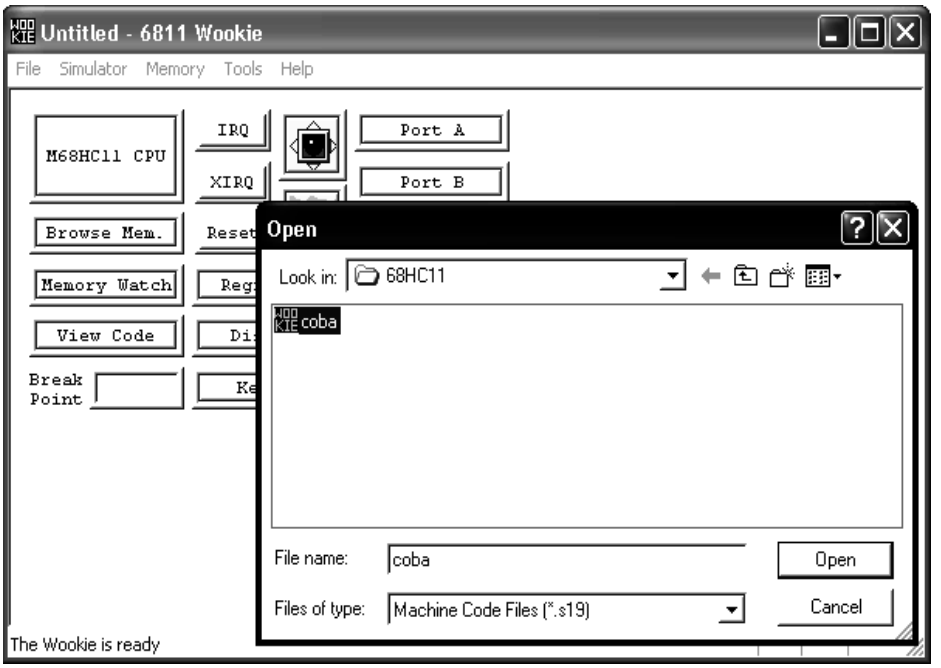
b. Sampai tahap ini anda telah memiliki file program dalam bahasa assembly versi 68HC11 yang disimpan pada file COBA.ASM. dan file hasil assembly dan kompilasi yang berektensi *.lst dan *.s19.
Tahap berikutnya adalah menjalankan simulator Wookie.
Buka Simulator Wookie:



Gambar 11. Tampilan Simulator Wookie

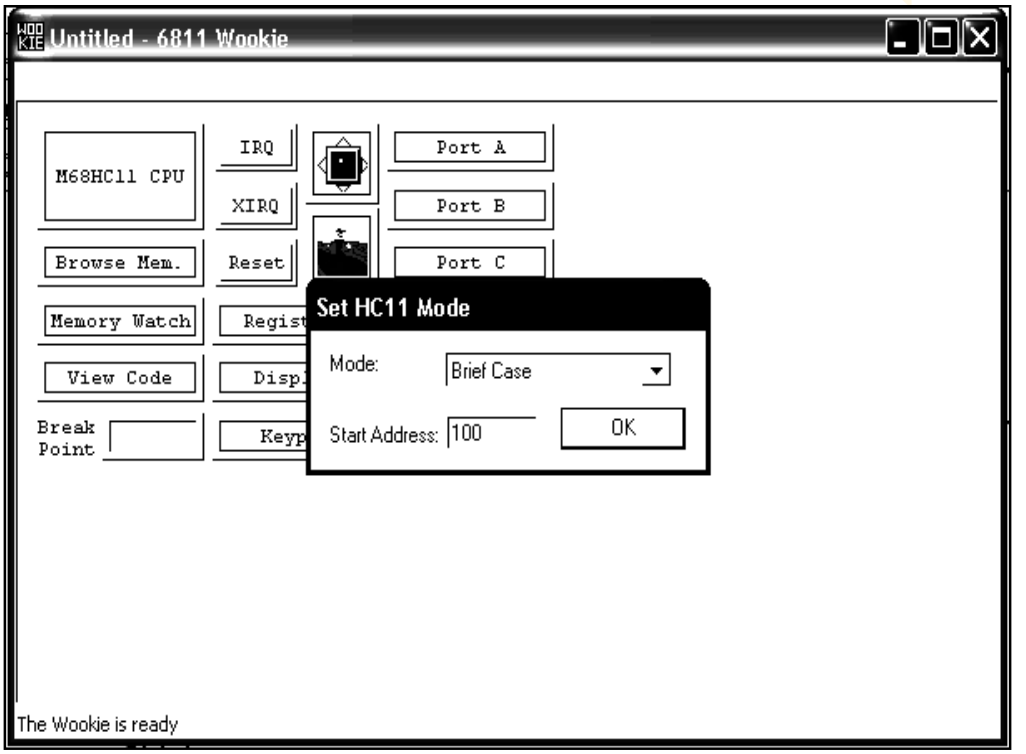
c. Lakukan loading file .S19 ke dalam simulator Wookie dengan click "File" pada menu kemudian click "Load .s19 file...". Pada jendela "Open", bukalah file .s19 yang baru saja

anda peroleh melalui proses kompilasi. Dalam hal ini anda pilih file COBA.S19. Sesuaikan dengan direktori anda menyimpan file.



Gambar 12. Tampilan Loading File .S19 Simulator Wookie

d. Setelah click pada “Open” akan muncul jendela "Set HC11 Mode". Pada jendela tersebut terdapat tiga pilihan yakni “Brief Case”, “Rug Warrior Bootstrap”, dan “Rug Warrior Expanded”, maksudnya adalah anda harus memilih mode operasi 68HC11 pada mode single chip (Brief Case), mode bootstrap (Rug Warrior Bootstrap) atau mode operasi expanded (Rug Warrior Expanded). Pada praktikum ini anda diminta memilih mode operasi single chip, jadi melalui jendela "Set HC11 Mode" pilihlah “Brief Case”. Pada jendela itu juga terdapat “Start Address” dengan nilai default C000, itu berarti secara otomatis Wookie memberikan nilai register PC (program counter) sebesar C000. Karena program yang telah anda tulis beralamat awal $0100, maka anda harus mengubah “Start Address” dari C000 menjadi 100.

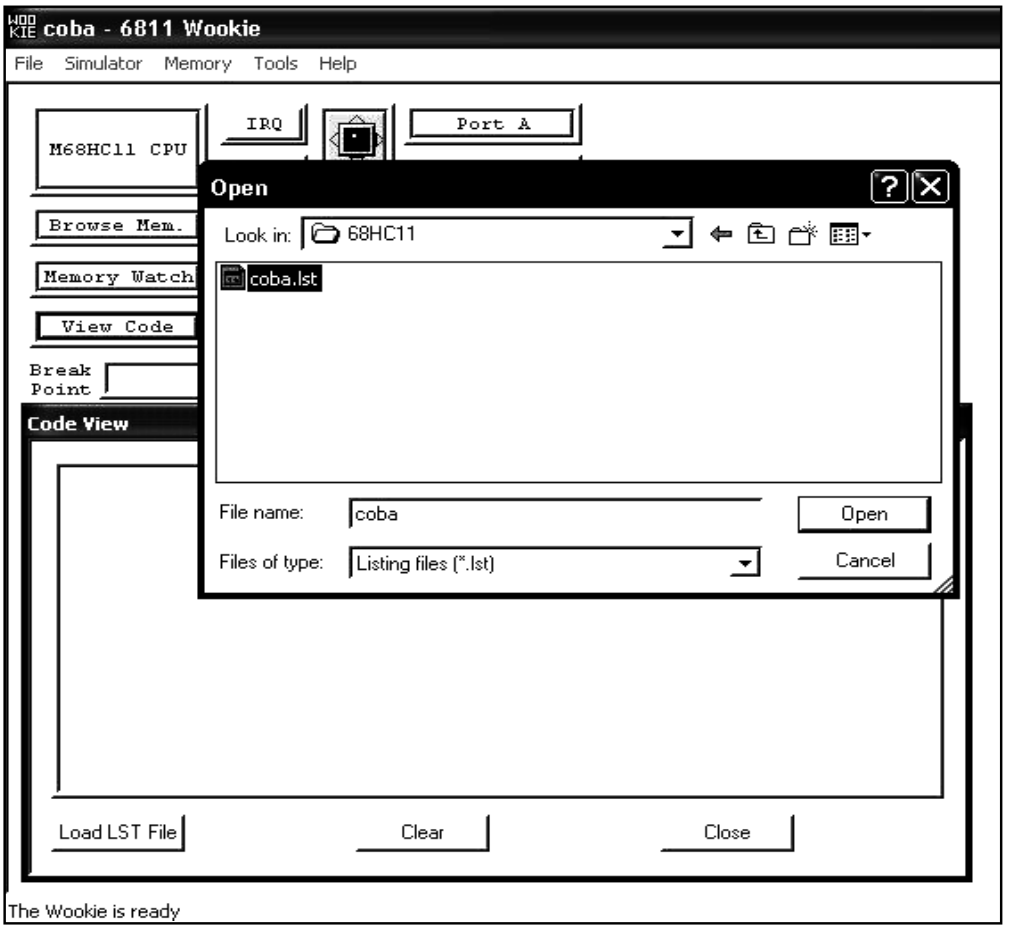


Gambar 13. Tampilan Mode Single Chip (Brief Case) Simulator Wookie

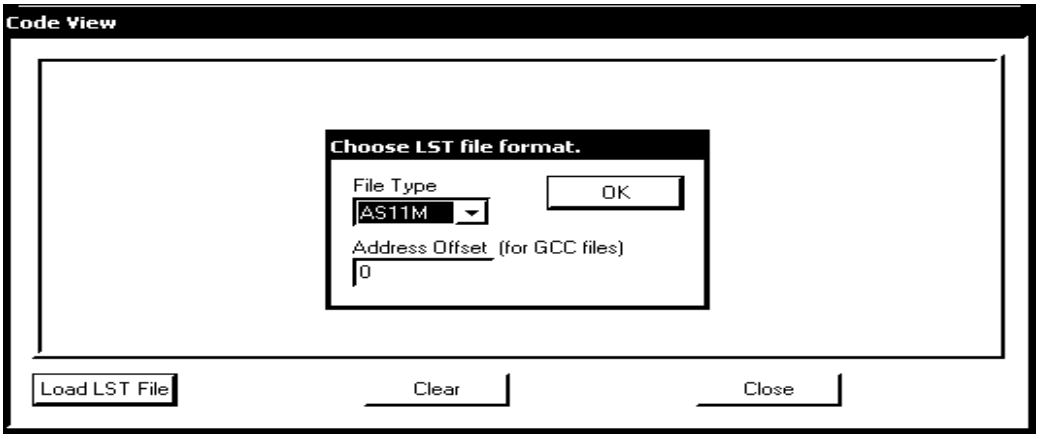
Jika click pada tombol “OK” tidak menimbulkan pesan kesalahan maka anda telah berhasil memuati simulator Wookie dengan file COBA.S19.

e. Selanjutnya anda harus menampilkan isi file .LST pada jendela “Code View”. Click tombol "View Code" dan melalui jendela Code View click tombol “Load LST File” untuk membuka file .LST. Dalam hal ini anda harus memilih terlebih dahulu jenis file yang akan dibuka. Pilih pada baris “Files of type” jenis file "Listing Files (*.lst)", kemudian pilih “File name” COBA.LST.

f. Jika penekanan tombol “Open” tidak menimbulkan pesan kesalahan, maka akan ditampilkan jendela “Choose LST file format”. Pada “File type” pilihlah “AS11M” kemudian click tombol OK. Jika tidak terjadi kesalahan, maka pada jendela “Code View” akan ditampilkan listing program yang terhimpun dalam file COBA.LST.

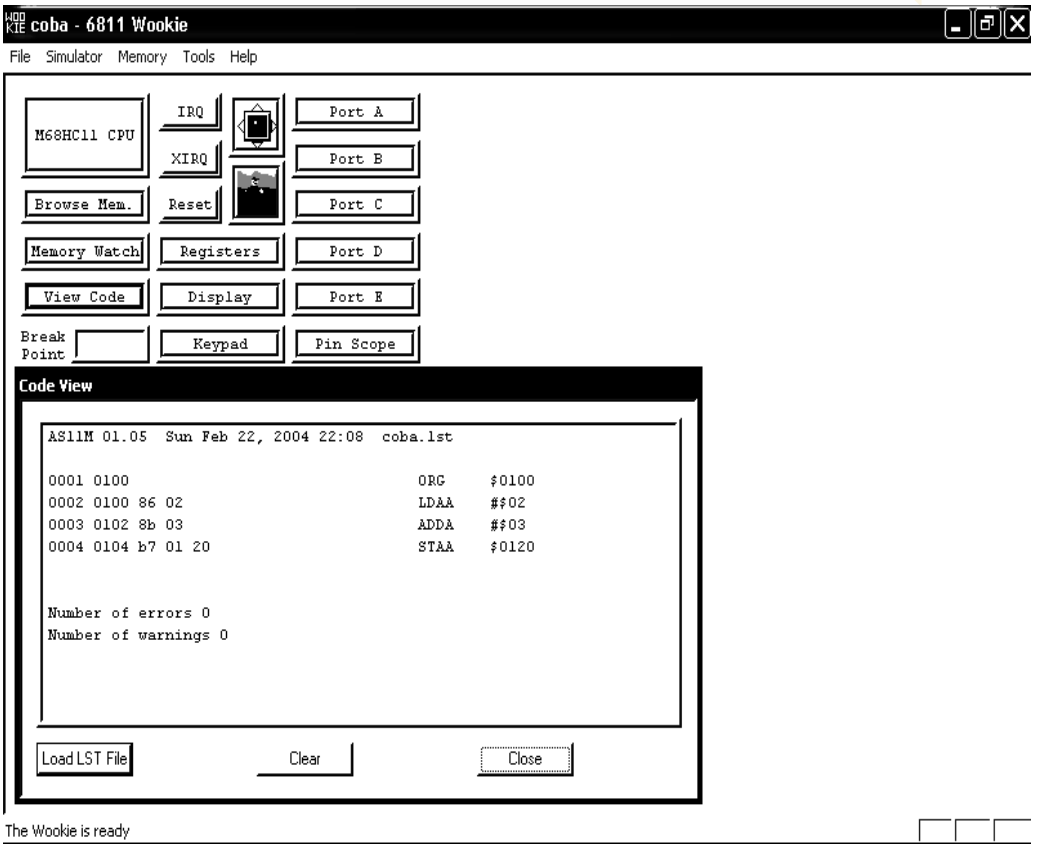


Gambar 14. Menampilkan Isi file .LST pada Jendela "Code View" Simulator Wookie



Gambar 15. Tampilan Pemilihan Jenis File "AS11M" Simulator Wookie

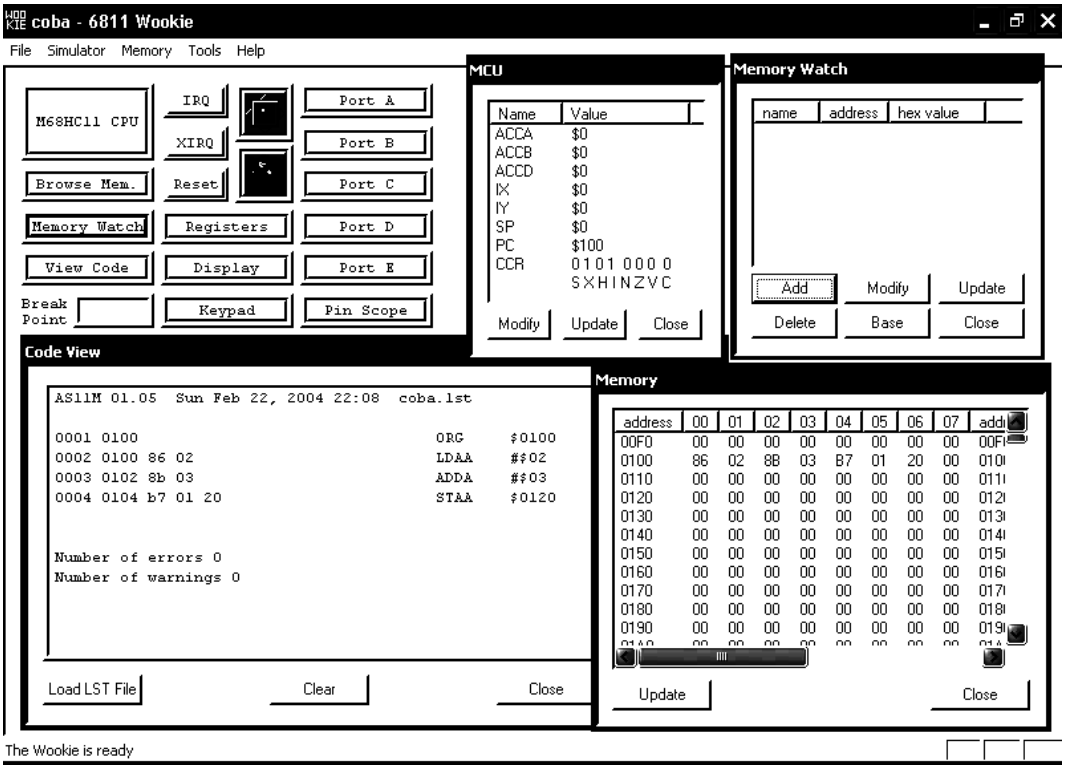
Jendela "Code View" menampilkan program listing sebagai berikut:



Gambar 16. Tampilan Jendela "Code View" yang Menampilkan Program Listing

g. Tampilan di atas menunjukkan bahwa Simulator Wookie telah memperoleh masukan file .S19 (dalam hal ini adalah COBA.S19) dan file .LST (dalam hal ini COBA.LST). Dalam keadaan seperti ini, simulator Wookie dapat dianggap seolah-olah berperan seperti sistem 68HC11 sesungguhnya yang memori internalnya telah diisi dengan instruksi-instruksi mesin yang terhimpun dalam file COBA.S19. Pada tahap ini anda sudah dapat menjalankan sistem 68HC11 tersebut untuk menguji program yang telah anda masukkan ke dalam memorinya. Agar anda dapat melihat kode-kode mesin dari program yang dimasukkan ke dalam memori dan selama program dijalankan dapat dilakukan pemantauan terhadap keadaan register, dan memorinya, bukalah jendela "MCU" untuk memantau register dengan

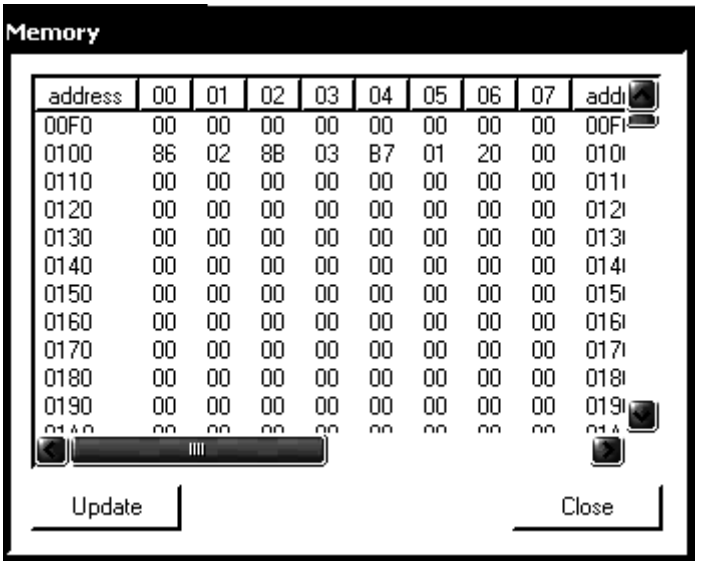
cara click tombol "M68HC11 CPU", bukalah jendela "Memory" dengan click tombol "Browse Mem.", dan bukalah jendela "Memory Watch" dengan click tombol "Memory Watch". Atur penempatan jendela-jendela tersebut agar anda dapat melakukan pemantauan resources 68HC11 dengan mudah selama program running. Contoh penempatan jendela-jendela pemantauan dapat dilihat pada gambar berikut ini:



Gambar 17. Tampilan Penempatan Jendela-jendela Pemantauan Terhadap Keadaan Register dan Memori

h. Sebelum menjalankan program, atur terlebih dahulu jendela-jendela pemantau. Atur jendela "Memory" agar dapat ditampilkan alamat awal $100. Ini dikarenakan program yang anda masukkan ke memori beralamat awal $100 sehingga kode-kode mesinnya akan ditempatkan mulai alamat $100. Pastikan hasil pengaturan jendela "Memory" sesuai gambar di bawah ini. Perhatikan isi alamat memori! Alamat $100 berisi kode 86 yang merupakan kode mesin dari LDAA, alamat $101 berisi kode 02 yang merupakan operand instruksi baris pertama,

dan seterusnya. Perhatikan bahwa isi memori juga dapat dibaca melalui listing program yang terdapat pada jendela "Code View". Coba anda tuliskan isi alamat memori dari $100 sampai dengan $106 berdasarkan listing program pada jendela "Code View" dan bandingkan dengan isi alamat yang sama melalui jendela "Memory"!

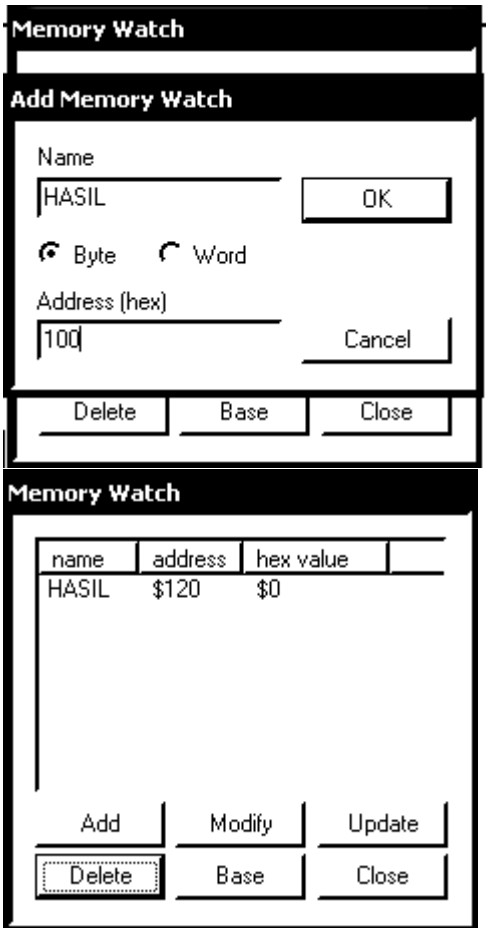


Gambar 18. Tampilan Pengaturan Jendela "Memory"

i. Untuk memantau keadaan memori selama program dijalankan, anda perlu mengatur jendela "Memory Watch". Ingat program anda dalam format assembly adalah:

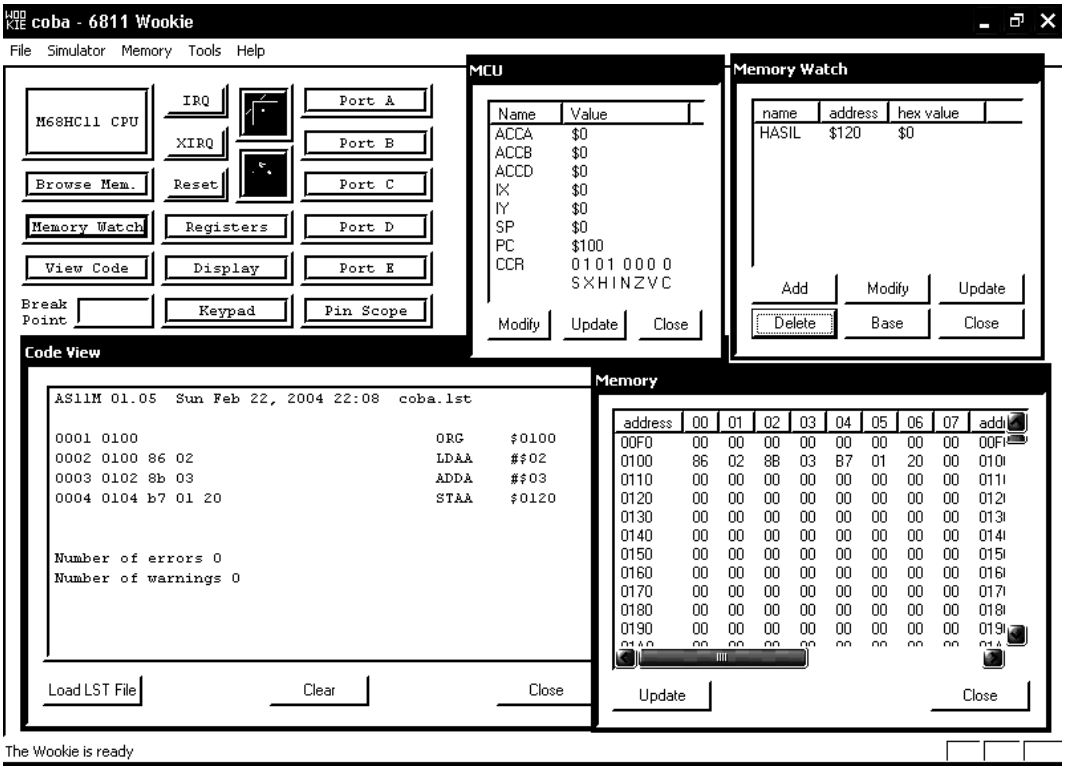
```
ORG     $0100
LDAA    #$2
ADDA    #$3
STAA    $0120
```

Dari program tersebut terlihat bahwa 68HC11 akan melakukan penjumlahan dua buah bilangan dan hasilnya disimpan ke alamat memori $120. Agar selama program dijalankan dapat dipantau isi alamat $120 sebagai penyimpan hasil, lakukan pengaturan jendela "Memory Watch" dengan click tombol "Add", isilah pada baris "Name" dengan HASIL dan isilah pada baris "Address (hex)" dengan 100 kemudian click tombol OK. Jika langkah anda benar akan muncul tampilan sebagai berikut:



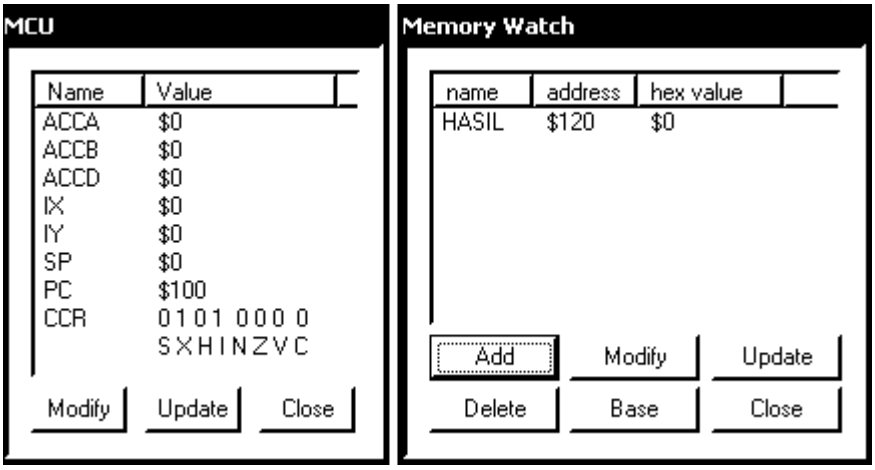
Gambar 19. Tampilan Pengaturan Jendela "Memory Watch"

Sampai tahap ini anda dapat dianggap telah berhadapan dengan sebuah sistem 68HC11 yang di dalam memorinya telah terdapat instruksi dalam format mesin hasil kompilasi program assembly dan dilengkapi dengan pemantau register dan memori. Pastikan tampilan akhir layar monitor anda adalah sebagai berikut:



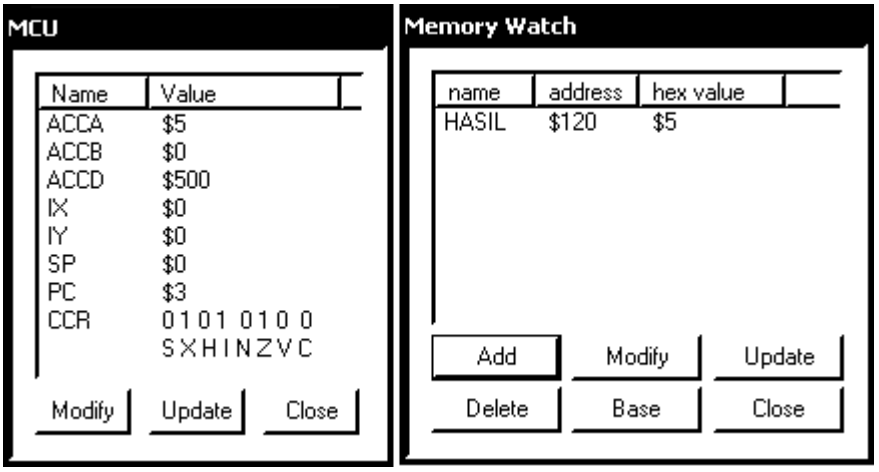
Gambar 20. Tampilan Akhir Layar Simulator Wookie

j. Running/testing Program Secara Menyeluruh Untuk menjalankan program, Wookie menyediakan 2 pilihan yakni program dijalankan secara keseluruhan, dan program dijalankan baris per baris (trace). Sebelum program dijalankan, amati dan catat isi akumulator ACCA, isi PC, dan isi alamat memori $120. Pastikan bahwa ACCA, alamat memori $120 keduanya berisi data 0, dan PC berisi $100 seperti terlihat pada tampilan berikut ini.



Gambar 21. Tampilan Isi Akumulator ACCA, Isi PC, dan Isi Alamat Memori $120

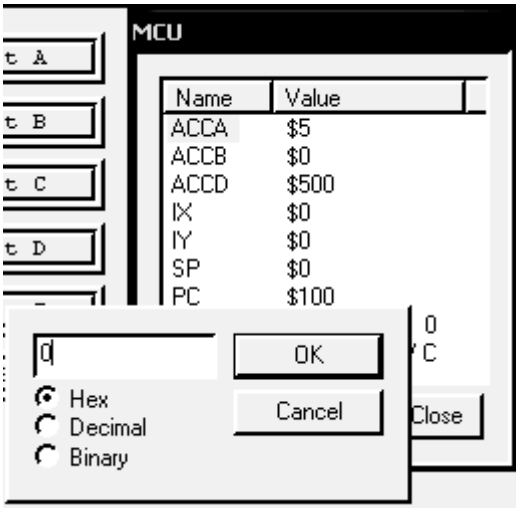
Jalankan program secara keseluruhan dengan menekan tombol merah sekali, sesaat kemudian ditekan sekali lagi. Dengan cara itu anda telah menjalankan program pada sistem 68HC11 secara keseluruhan. Amati isi ACCA, isi PC dan alamat memori $120, apakah ACCA dan alamat memori $120 telah berisi $5? Jika ya maka program anda telah dapat berjalan dengan baik pada system 68HC11. Setelah program dijalankan, pastikan bahwa tampilan jendela "MCU" dan "Memory Watch" sebagai berikut:



Gambar 22. Tampilan Jendela "MCU" dan "Memory Watch" Setelah Program Dijalankan

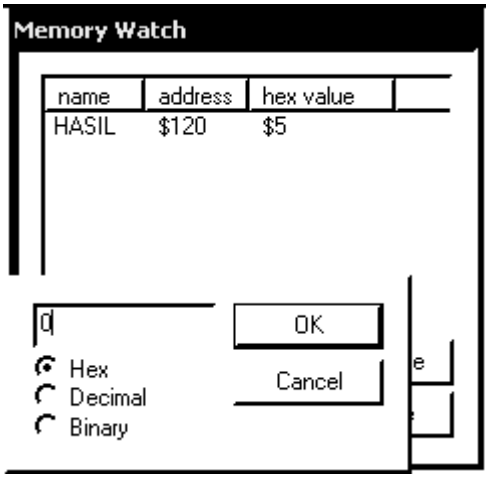
k. Running/testing Program Setiap Baris (trace)

Lakukan reset terlebih dahulu terhadap sistem 68HC11 dengan click tombol "Reset" pada jendela utama Wookie. Tujuan reset untuk mengembalikan isi PC ke keadaan awal berisi alamat awal program yakni $100. Selanjutnya lakukan modifikasi terhadap isi ACCA pada jendela "MCU" dengan click ACCA dan tombol "Modify" pada jendela "MCU", isilah dengan 0 untuk ACCA dan click OK.



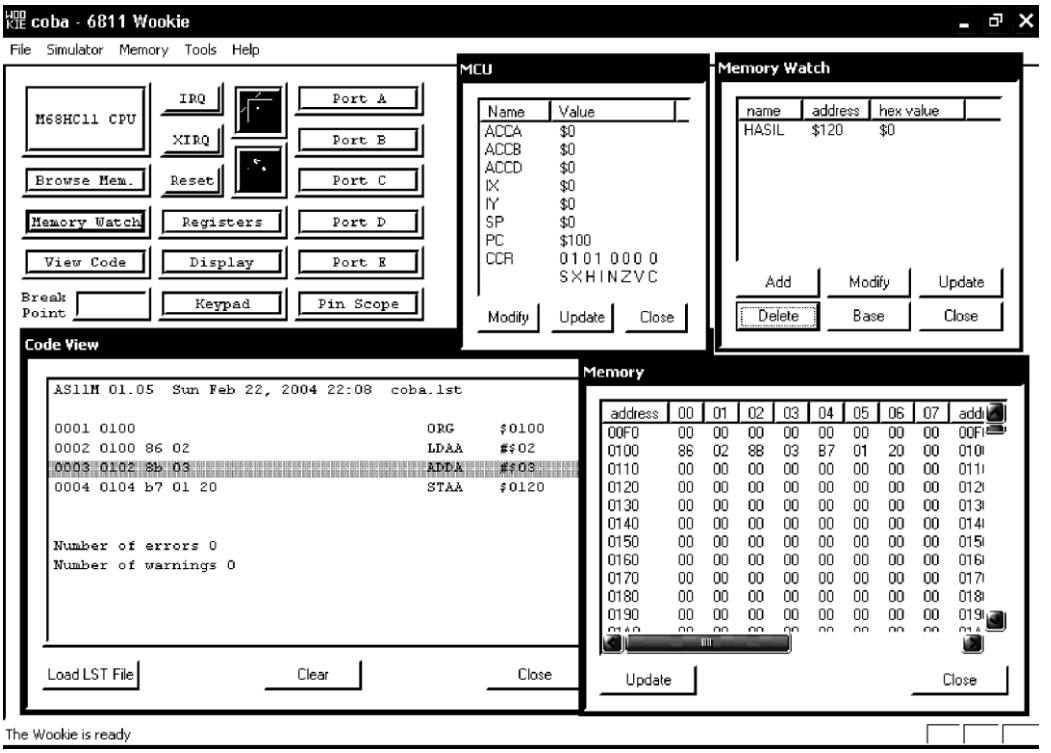
Gambar 23. Tampilan Reset pada Jendela Utama Wookie

Lakukan pula modifikasi terhadap isi HASIL dengan click HASIL dan tombol "Modify" pada jendela "Memory Watch" kemudian isi dengan 0 dan click OK.



Gambar 24. Tampilan Modifikasi Terhadap Isi HASIL dengan Click HASIL dan Tombol "Modify" pada Jendela "Memory Watch"

Sebelum melakukan trace yakni menjalankan program baris demi baris, pastikan bahwa jendela-jendela pemantau telah diatur sehingga isi PC=$100 (alamat awal program anda), isi ACCA=$0, dan isi alamat $120 adalah $0 seperti tampilan berikut ini:



Gambar 25. Tampilan Menjalankan Program Baris Demi Baris

Untuk menjalankan program baris demi baris pada jendela utama Wookie tekan tombol berwarna hijau atau tombol dengan gambar orang sedang berjalan. Penekanan tombol sekali berarti anda telah menjalankan baris program yang alamatnya ditunjukkan oleh isi PC. Perhatikan pada jendela "Code View" bahwa saat pertama kali anda menekan tombol hijau, baris program 0002 tereksekusi dan nilai PC mengalami increment sehingga menunjuk baris berikutnya yang siap dijalankan. Baris berikutnya yang siap dijalankan adalah baris program 0003, perhatikan bahwa setiap baris program yang akan dieksekusi akan disorot (highlight). Jalankan program anda baris demi baris sampai selesai dan setiap anda menekan tombol hijau, amati isi ACCA pada jendela "MCU" , dan isi alamat $120 pada jendela "Memory Watch".

l. Debugging

Jika anda ingin mengubah program yang semula:

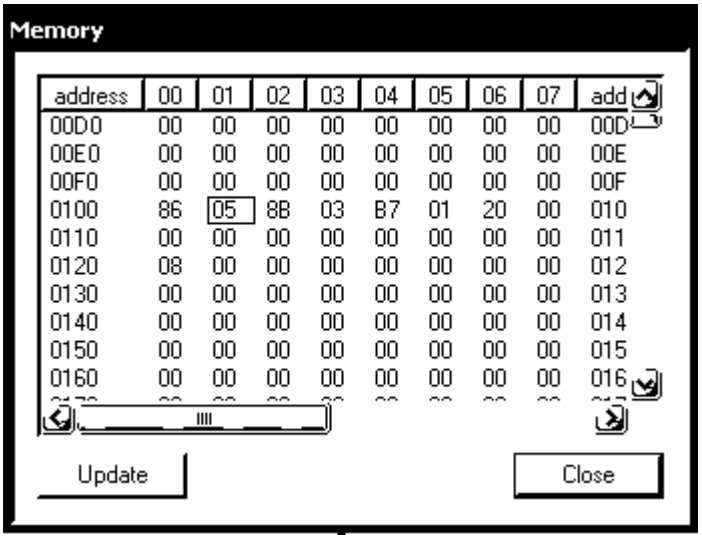
```
ORG     $0100
LDAA    #$2
ADDA    #$3
STAA    $0120
```

misalnya menjadi:

```
ORG     $0100
LDAA    #$5
ADDA    #$3
STAA    $0120
```

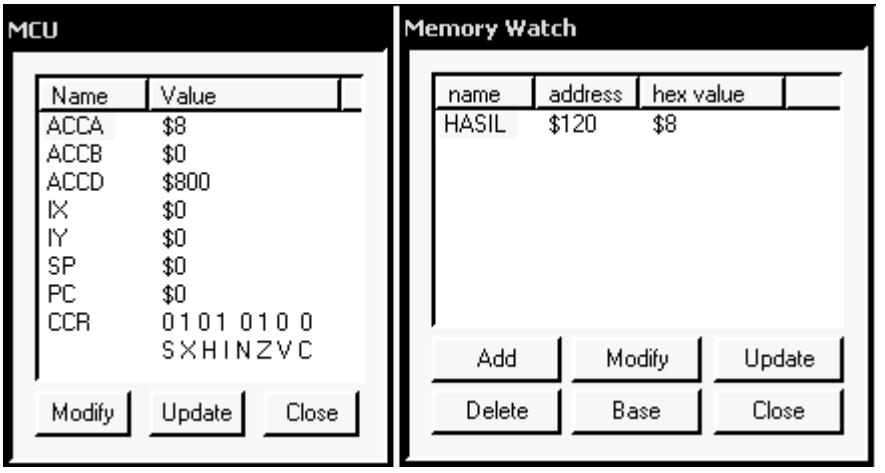
maka anda dapat melakukannya dengan cara biasa yakni perbaiki program tersebut dengan editor Q.EXE, lakukan kompilasi dengan AS11M.EXE, kemudian load kembali ke dalam memori 68HC11. Namun, cara itu tidak efisien sehingga diperlukan cara tertentu agar anda memperoleh kemudahan dalam menguji program. Wookie menyediakan fasilitas debugging program yang akan mempermudah kerja anda dalam menguji program. Jika anda akan mengubah operand pada baris ke-2 dari $2

menjadi $5, maka anda cukup mengubah secara langsung isi alamat memori tempat operand tersebut diletakkan dengan data 05. Perhatikan pada jendela "Code View" bahwa operand 02 menempati alamat memori $101, sehingga untuk mengubah operand 02 dengan 05, melalui jendela "Memory" double click pada alamat $101 tulislah 05 kemudian click pada tombol "Update", pastikan anda memperoleh tampilan berikut ini.



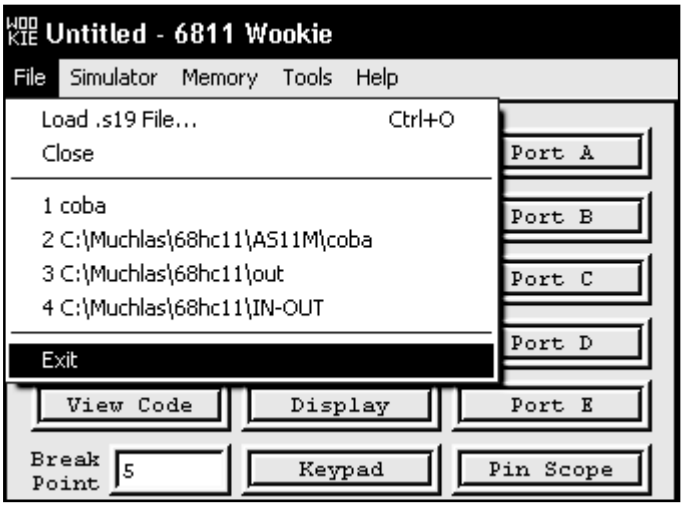
Gambar 26. Tampilan Menjalankan Debugging Program

Dengan cara seperti itu, maka anda telah melakukan perubahan program langsung terhadap kode-kode mesinnya, dan sekarang program tersebut siap untuk dijalankan. Perlu dicatat bahwa debugging dengan cara ini hanya menyebabkan perubahan pada isi memori tetapi tidak akan mengubah listing program yang ada pada jendela "Code View". Sebelum anda menjalankan program yang telah diubah tersebut, lakukan terlebih dahulu reset, dan modifikasi pada ACCA dan HASIL sehingga PC=$100, ACCA=$0 dan HASIL=$0. Jalankan program secara keseluruhan atau baris demi baris, pastikan bahwa setelah seluruh baris program dijalankan anda akan memperoleh hasil sebagai berikut:



Gambar 27. Tampilan Hasil Reset, Modifikasi pada ACCA dan HASIL

m. Keluar dari Wookie
Untuk keluar Wookie, click File diteruskan Exit.



Gambar 28. Tampilan Keluar Dari Wookie

G. Tugas Akhir

1. Berdasarkan praktikum yang telah anda lakukan, program dalam format apakah yang disimpan dalam memori sistem 68HC11?
2. Tulislah kembali kode-kode mesin dari mnemonic LDAA, ADDA, dan STAA!

3. Perhatikan program berikut ini:

```
ORG     $0150
LDAA    #$05
SUBA    #$01
ADDA    #$03
STAA    $0170
```

   Jika kode mesin untuk mnemonic SUBA adalah 80 heksadesimal, susunlah peta memori yang menunjukkan alokasi kode-kode mesin dari program assembly tersebut! Gunakan format seperti pada jendela "Memory" simulator Wookie! Coba tulis listing program hasil kompilasi menurut perkiraan anda tanpa melakukan kompilasi yang sesungguhnya! Jika program tersebut dijalankan, tulislah isi ACCA, isi PC dan isi alamat $170 setiap suatu baris dieksekusi.
4. Tulis program untuk menampilkan bilangan $8A ke port B! Lakukan kompilasi terhadap program tersebut dan jalankan dengan menggunakan simulator Wookie! Lengkapi jendela utama Wookie dengan jendela "MCU", jendela "Register Watch", jendela "Code View", jendela "Memory", dan Jendela "Port B". Setelah anda menjalankan program pada sistem 68HC11 lewat Wookie, cetaklah tampilan akhir jendela Wookie yang menunjukkan hasil running program. Jika program tidak berhasil dijalankan, lakukan troubleshooting sehingga program anda berhasil dijalankan dengan Wookie.

BAB 3
PERCOBAAN II

Eksplorasi Sistem 68HC11 Dengan Komputer Host

A. Kompetensi Dasar

Mahasiswa mampu menggunakan EVBU M68HC11 dan BUFFALO

B. Indikator Pencapaian Kompetensi

1. Melakukan downloading kode-kode S-record dari komputer host ke sistem 68HC11
2. Melakukan eksplorasi resources sistem 68HC11 (register CPU, memori) dengan perintah-perintah BUFFALO melalui AxIDE
3. Running program pada sistem 68HC11 mode single chip melalui komputer host Mengkompilasi instruksi assembly menjadi S-record dengan compiler AxIDE
4. Tracing program pada sistem 68HC11
5. Debugging program pada sistem 68HC11 dengan perintah-perintah BUFFALO melalui AxIDE

C. Dasar Teori

Sistem-sistem berbasis mikrokontroler pada umumnya memiliki ciri-ciri kompak dalam arti berukuran minimal tetapi memberikan fungsi yang maksimal. Hal itu bisa tercapai karena mikrokontroler merupakan suatu piranti elektronik yang bersifat on-chip-memory dan bahkan on-chip-peripheral, artinya di dalam chip mikrokontroler yang berukuran kecil tersebut telah terkandung suatu sistem interkoneksi yang terdiri atas mikroprosesor, read only memory (ROM), random access memory (RAM), dan berbagai peripheral yang dibutuhkan dalam suatu perancangan sistem. Untuk menghasilkan mikrokontroler yang dapat bekerja mendukung sistem tersebut, mikrokontroler harus diprogram terlebih dahulu sehingga memorinya berisi instruksi-instruksi sesuai dengan kebutuhan sistem.

Tahap-tahap pemrograman mikrokontroler cukup panjang, dimulai dengan penulisan instruksi dalam bahasa assembly, selanjutnya instruksi tersebut diubah ke dalam format bahasa mesin (heksadesimal),

instruksi yang telah berbentuk heksadesimal tersebut kemudian ditempatkan di dalam memori mikrokontroler, dan akhirnya baru dapat dijalankan. Tahap-tahap tersebut telah anda pelajari pada percobaan yang lalu dengan menggunakan simulator Wookie sebagai mikrokontroler 68HC11 yang bersifat virtual (maya). Dalam kenyataannya, yakni ketika anda menggunakan sistem 68HC11 yang sesungguhnya, program yang telah dikompilasi menjadi bahasa mesin (S-Record) itu dimasukkan ke dalam memori sistem 68HC11 menggunakan prosedur downloading tertentu. Salah satu cara adalah dengan menghubungkan komputer host berbentuk personal computer (PC) dengan sistem 68HC11.

Dalam hal ini peran komputer host selain sebagai pengirim kode S-record ke memori 68HC11 (sebagai downloader), juga dapat digunakan sebagai fasilitas untuk menulis instruksi assembly, sebagai fasilitas mengubah instruksi assembly ke format heksadesimal (mesin), dan melalui komputer host instruksi-instruksi yang ada pada sistem 68HC11 dapat dijalankan. Selain itu, melalui komputer host dapat dilakukan debugging untuk memperbaiki instruksi-instruksi yang ada di dalam memori sistem 68HC11. Hal itu dapat dimungkinkan karena saat ini telah diproduksi oleh MOTOROLA sistem 68HC11 versi EVBU (evaluation board unit) yang di dalamnya telah disediakan fasilitas komunikasi standar RS232 dan secara built-in di dalam memorinya (ROM) telah diletakkan program komunikasi dengan komputer host yang dinamakan BUFFALO (Bit User Fast Friendly Aid to Logical Operation). Melalui fasilitas komunikasi tersebut maka sistem 68HC11 dapat berkomunikasi dengan komputer host sehingga mikrokontroler tersebut dapat dieksplorasi, dieksploitasi dan dipelajari dengan mudah.

D. Tugas Pendahuluan

1. Jelaskan cara melakukan pengaturan hardware (jumper) sistem 68HC11EVBU agar beroperasi pada mode single chip! Jelaskan pula pengaturannya agar bekerja pada mode expanded!
2. Gambarkan peta memori sistem 68HC11 untuk berbagai mode pengoperasian!
3. Pada mode single chip, berapa kapasitas RAM internal sistem 68HC11 yang dapat diakses oleh pemrogram? Menurut peta memori, alamat RAM tersebut dialokasikan pada alamat berapa saja?

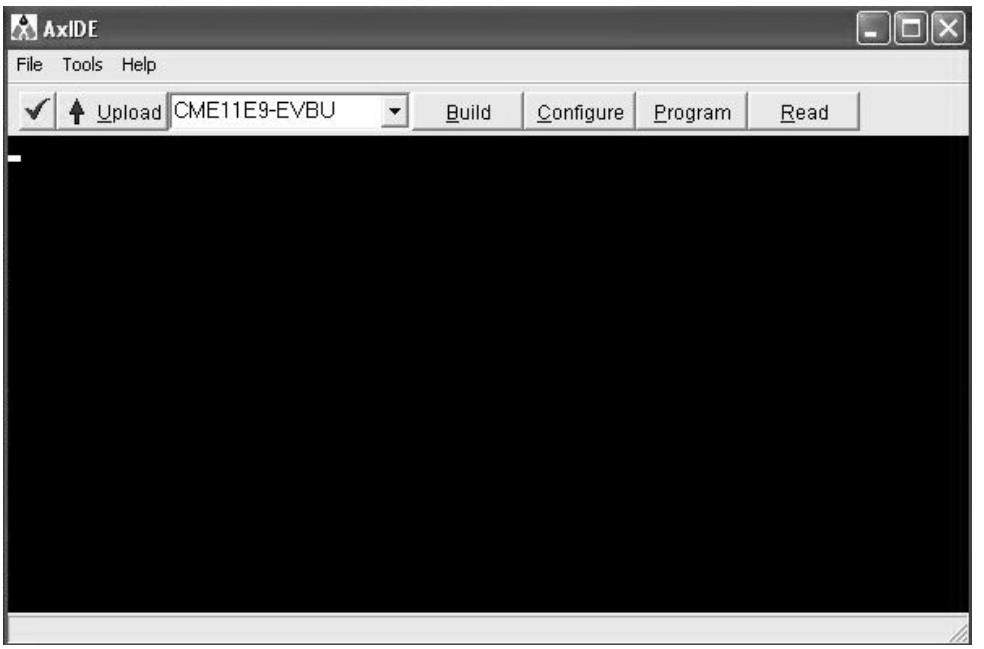
4. Menurut Reference Manual dari sistem 68HC11, interkoneksi sistem pada mode expanded dapat menyediakan RAM eksternal dan EPROM eksternal berkapasitas berapa? Berdasarkan rangkaian dekoder alamatnya, RAM dan EPROM eksternal tersebut dialokasikan pada alamat berapa saja?
5. Gambarkan instalasi komputer host dengan sistem 68HC11EVBU! Dalam interkoneksi tersebut jelaskan fungsi komputer host!
6. Apa yang dimaksud dengan program monitor BUFFALO yang terdapat di dalam ROM sistem 68HC11EVBU, dan apa fungsinya?
7. Jelaskan pengaturan jumper pada board sistem 68HC11EVBU agar melalui komputer host, program monitor BUFFALO dapat dijalankan dengan baik
8. Tulislah perintah program monitor BUFFALO untuk:
   a. download kode-kode mesin dalam file .S19 dari komputer host ke memori sistem 68HC11EVBU!
   b. melihat isi alamat memori $100 sampai dengan $200
   c. mengubah isi alamat memori $110 sampai dengan $115 dengan data $2A
   d. mengubah isi alamat memori $120 dengan data $1B
   e. melihat isi register program counter (PC), akumulator A, akumulator B, indeks X, indeks Y, register kondisi (flags), dan stack pointer!
   f. mengubah isi register PC dengan data $100
   g. mengubah isi akumulator A dengan data $00

E. Alat-alat dan Langkah Percobaan

1. Satu unit sistem 68HC11 EVBU
2. Satu unit modul Basic I/O
3. Satu unit catu daya
4. Software AxIDE
5. Software Programmer's Notepad

F. Persiapan Percobaan

1. Pasang Kabel Serial Port COM pada Host PC.
2. Atur posisi jumper pada EVBU agar beroperasi pada mode single-chip dan Trace enable, yaitu dengan memasang jumper J3 dan J6.
3. Hubungkan Kabel Serial yang dengan EVBU.
4. Hubungkan Catu Daya dengan EVBU.
5. Aktifkan/Run program AxIDE.



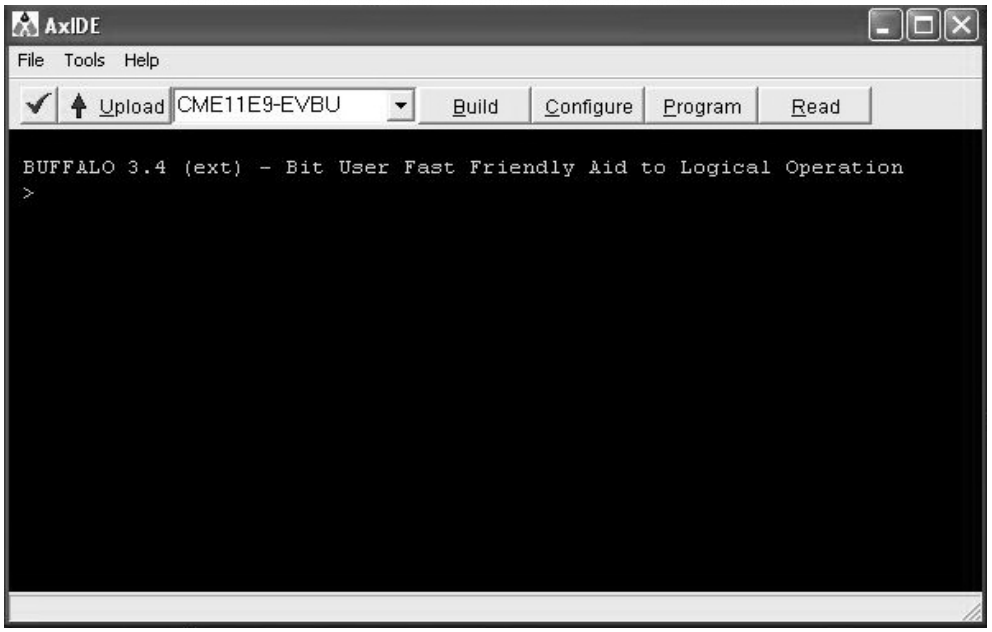
Gambar 29. Tampilan Program AxIDE

6. Click icon Check (Centang) atau melalui menu File>Option, pastikan setting komunikasi yang digunakan adalah sebagai berikut :

Tabel 5. Setting Komunikasi Program AxIDE

Port	COM1/COM2	Handshaking	Information
Baud Rate	9600		
Parity	None	Xon/Xoff	OFF
Data Bits	8	Rts/Cts	OFF
Stop Bits	1	Dtr/Dsr	OFF

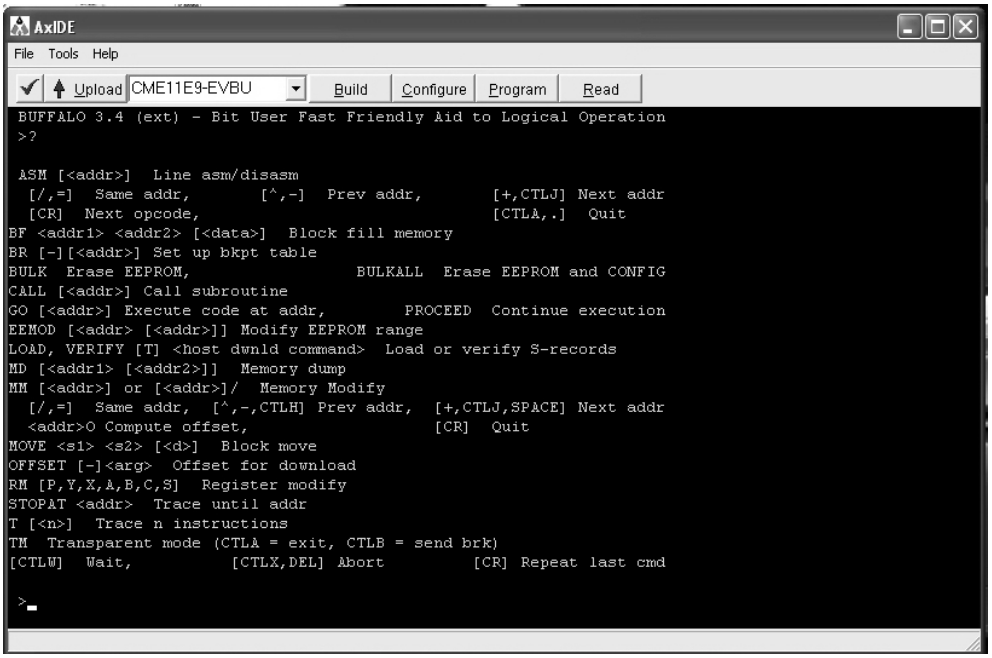
   Jika belum sesuai lakukan pengaturan agar diperoleh setting seperti tampak pada tabel di atas.
7. Tekan tombol Reset pada EVBU.
8. Pastikan muncul tampilan BUFFALO pada terminal window AxIDE, yang menunjukkan bahwa EVBU dan komputer telah terhubung melalui Port Serial.
9. Tekan Enter agar anda berada pada prompt BUFFALO sebagai berikut:



Gambar 30. Tampilan Prompt BUFFALO Program AxIDE

Keterangan lebih lengkap ada pada petunjuk penggunaan software halaman 01 s/d 08.

10. Menu bantuan perintah-perintah BUFFALO dapat ditampilkan dengan mengetik perintah:
    >? diikuti ENTER, sehingga akan tampil:

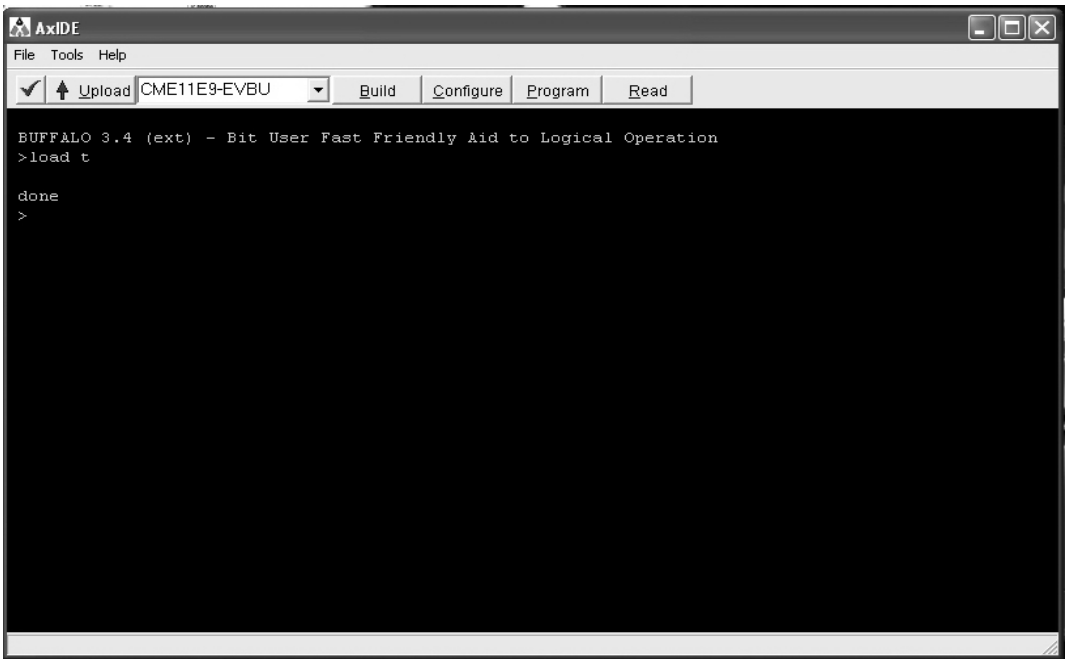


Gambar 31. Tampilan Menu Bantuan Perintah-perintah BUFFALO

11. Menggunakan software Programmer's Notepad, tulis program berikut ini:

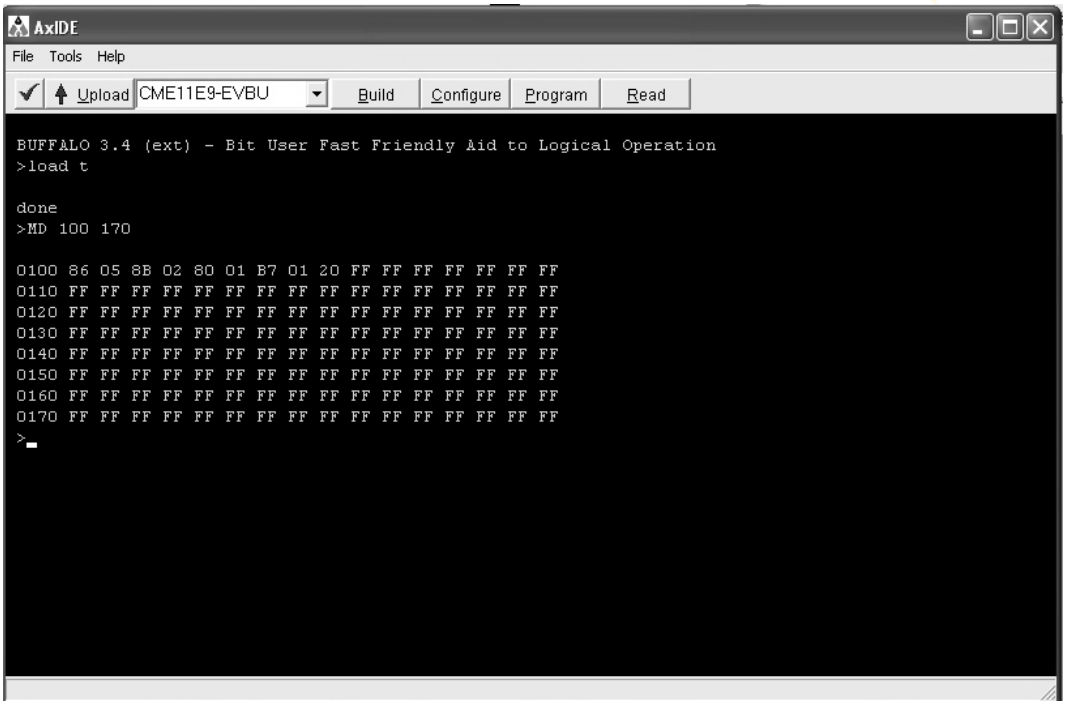
```
ORG     $0100   Alamat awal program $0100
LDAA    #$05    Memuati ACCA dengan $5
ADDA    #$02    Menambahkan ke ACCA $2
SUBA    #$01    Mengurangkan ACCA dengan $1
STAA    $0120   Simpan isi ACCA ke alamat $0120
```

Simpan dengan nama COBA2.asm. Lakukan assembly melalui program AxIDE, Download hasil assembly ke EVBU. Jika proses download itu berhasil maka anda akan memperoleh tampilan sebagai berikut.



Gambar 32. Tampilan Download Hasil Assembly ke EVBU

12. Melihat isi memori sistem 68HC11.
Tampilan di atas menunjukkan bahwa memori internal dalam hal ini adalah random access memory (RAM) dari sistem 68HC11 telah dimuati dengan kode-kode mesin yang terhimpun di dalam file COBA2.S19. Untuk melihat kode-kode mesin itu berikan perintah **MD**. Karena program anda dimulai pada alamat $100, maka berikan perintah MD 100 170, sehingga akan tampil:



Gambar 33. Tampilan Kode-kode Mesin yang Terhimpun pada File

Coba bandingkan tampilan isi memori sistem 68HC11 yang sesungguhnya dengan tampilan isi memori sistem 68HC11 dari simulator Wookie, sama bukan? Apa makna kode-kode mesin itu? Ingat! Program assembly anda adalah:

```
ORG     $0100   Alamat awal program $0100
LDAA    #$05    Memuati ACCA dengan $5
ADDA    #$02    Menambahkan ke ACCA $2
SUBA    #$01    Mengurangkan ACCA dengan $1
STAA    $0120   Simpan isi ACCA ke alamat $0120
```

dan isi file listing program adalah:

```
0001 0100                ORG     $0100   Alamat awal program $0100
0002 0100 86 05          LDAA    #$05    Memuati ACCA dengan $5
0003 0102 8b 02          ADDA    #$02    Menambahkan ke ACCA $2
0004 0104 80 01          SUBA    #$01    Mengurangkan ACCA dengan $1
0005 0106 b7 01 20       STAA    $0120   Simpan isi ACCA ke alamat $0120
0006
```

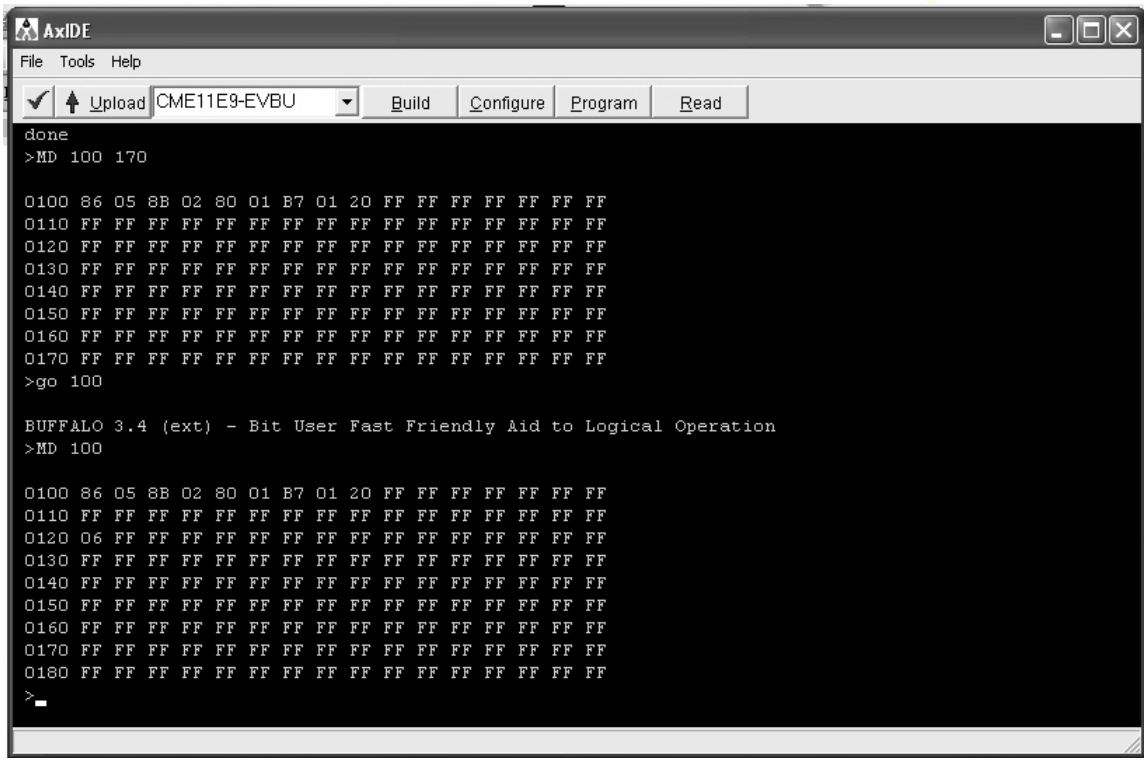
Jadi, makna tabel memori yang ditampilkan dengan perintah MD di atas adalah:

Alamat	$100	berisi kode mesin 86 kode dari mnemonic	LDAA
Alamat	$101	berisi kode mesin 05 kode dari operand	#$05
Alamat	$102	berisi kode mesin 8B kode dari mnemonic	ADDA
Alamat	$103	berisi kode mesin 02 kode dari operand	#$02
Alamat	$104	berisi kode mesin 80 kode dari mnemonic	SUBA
Alamat	$105	berisi kode mesin 01 kode dari operand	#$01
Alamat	$106	berisi kode mesin B7 kode dari mnemonic	STAA
Alamat	$107	berisi kode mesin 01 kode dari operand	$0120
Alamat	$108	berisi kode mesin 20 kode dari operand	$0120

Dengan demikian, total program anda menempati 9 byte, kode-kode mesin itu disimpan dalam RAM sistem 68HC11 yakni dari alamat $100 sampai dengan alamat $108. Khusus untuk operand dengan panjang 2 byte, kode- kode mesinnya menempati alamat 2 lokasi, dalam hal ini operand $0120 ditempatkan pada alamat $107 untuk byte orde tinggi 01 dan alamat $108 untuk byte orde rendah 20.

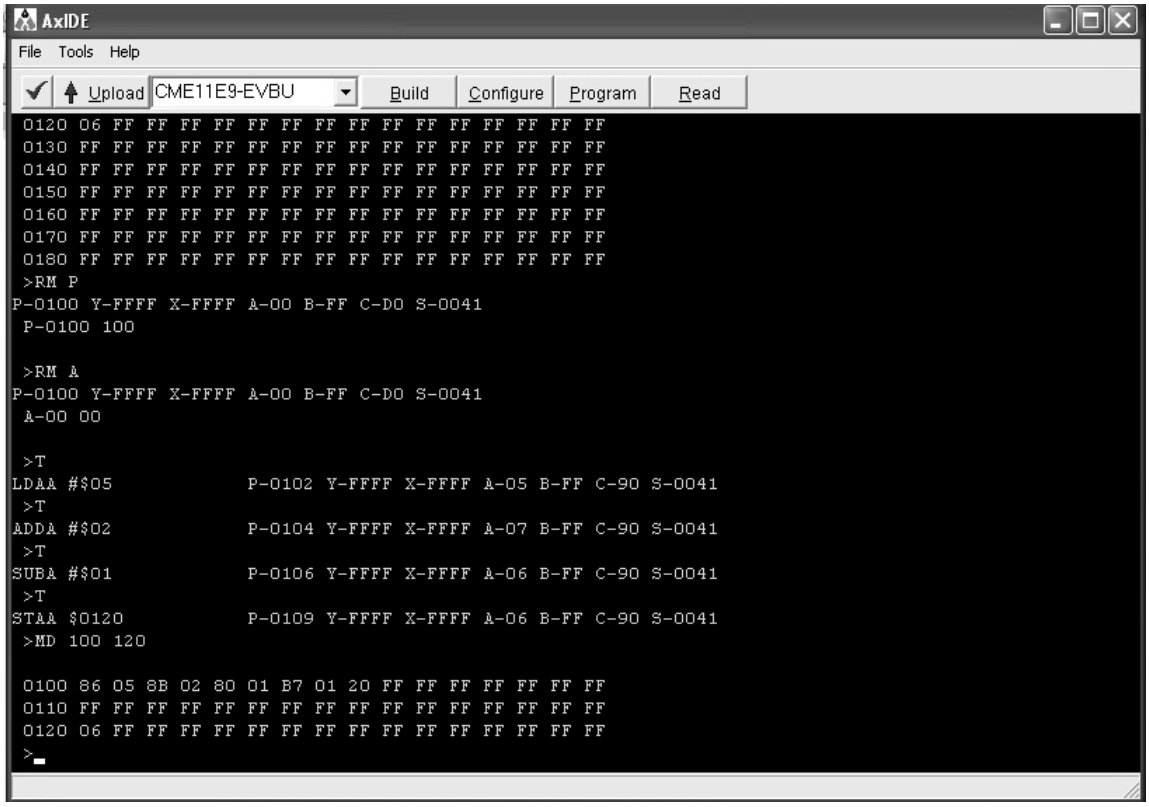
13. Running Program Keseluruhan.

Setelah RAM sistem 68HC11 dimuati dengan kode-kode mesin .S19, maka program tersebut dapat dijalankan. Sebelum menjalankan program, amati/catat isi alamat $120 yang merupakan alamat untuk menyimpan hasil operasi $5+$2-$1, dalam hal ini berisi FF bukan?. BUFFALO menyediakan perintah running program dalam dua mode, yakni running program keseluruhan, dan running program baris per baris (trace). Pada tahap ini anda akan menjalankan keseluruhan baris program. Berikan AxIDE perintah **GO 100** diikuti ENTER dan sesaat kemudian tekanlah tombol **RESET** pada board sistem 68HC11. Selanjutnya lihatlah isi memori alamat $120 dengam perintah **MD 100** diikuti menekan ENTER sehingga diperoleh tampilan seperti berikut:



Gambar 34. Tampilan Keseluruhan Baris Program AxIDE

14. Menjalankan program baris demi baris (trace).
    Sekarang anda akan berlatih menjalankan program baris per baris (trace). Sebelumnya, atur terlebih dahulu register PC (program counter) agar berisi $100 sesuai dengan alamat awal program anda, berikan perintah **RM P** tekan ENTER kemudian ikuti dengan mengetik **100** dan tekan ENTER. Ubah pula isi akumulator A dengan data 00, lakukan dengan perintah **RM A** tekan ENTER diteruskan dengan mengetik 00 dan tekan ENTER. Untuk menjalankan program per baris berikan perintah **T** diikuti ENTER. Berikan perintah **T** sebanyak empat kali, dan diakhiri dengan perintah **MD 100 120** tekan ENTER untuk melihat isi memori alamat $120. Pastikan hasilnya sesuai dengan tampilan berikut ini.



Gambar 35. Tampilan Program Baris per Baris (Trace)

Dari tampilan di atas terlihat bahwa setiap trace yang dilakukan kita dapat melihat isi semua register sebagai dampak eksekusi suatu baris program. Coba catatlah isi register PC dan akumulator A setiap trace yang dilakukan! Pada trace ke-4, apakah isi akumulator A sama dengan isi alamat $120 yang diperoleh melalui running keseluruhan maupun melalui trace? Sama bukan?

15. Debugging.

Program monitor BUFFALO juga memungkinkan kita melakukan perbaikan program pada level kode mesin (debugging). Prinsip perbaikan program di sini adalah pengubahan isi memori dengan suatu perintah memory modify (MM). Anggap bahwa program yang sudah anda tulis yakni:

```
ORG     $0100   Alamat awal program $0100
LDAA    #$05    Memuati ACCA dengan $5
ADDA    #$02    Menambahkan ke ACCA $2
SUBA    #$01    Mengurangkan ACCA dengan $1
STAA    $0120   Simpan isi ACCA ke alamat $0120
```

akan diubah menjadi:

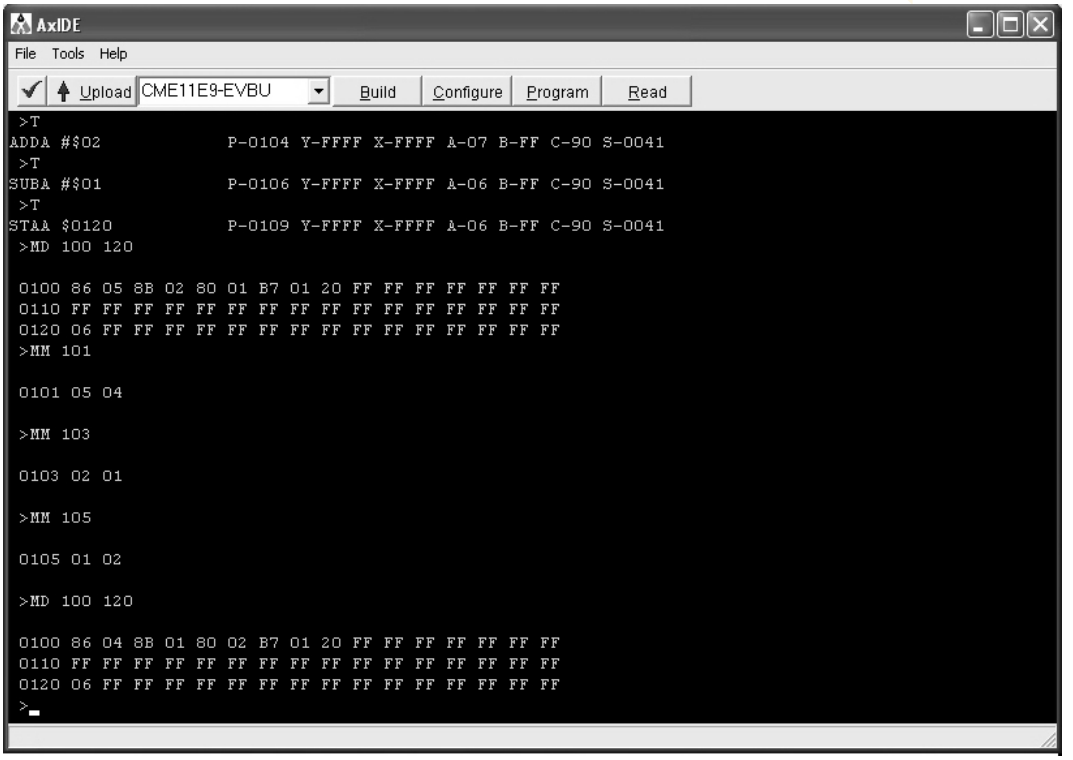
```
ORG     $0100   Alamat awal program $0100
LDAA    #$04    Memuati ACCA dengan $4
ADDA    #$01    Menambahkan ke ACCA $1
SUBA    #$02    Mengurangkan ACCA dengan $2
STAA    $0120   Simpan isi ACCA ke alamat $0120
```

Pengubahan program dapat dilakukan dengan cara biasa yakni diubah dulu program assembly nya dengan Programmer's Notepad, dikompilasi dengan dan download file .S19 hasil kompilasi dari komputer host ke RAM 68HC11 dengan AxIDE, baru dijalankan. Tetapi cara seperti tidak efisien sehingga perlu dilakukan debugging dengan cara mengubah langsung pada kode-kode mesin yang terdapat pada RAM 68HC11.

16. Mengubah Isi Memori

Dari tampilan isi memori mulai alamat $100 di atas terlihat bahwa kode mesin untuk operand #$05 baris program ke-2 menempati alamat $101, operand #$02 baris program ke-3 menempati alamat $103, dan operand #$01 baris program ke-4 menempati alamat $105. Dalam debugging ini anda akan mengubah isi alamat $101, $103, dan $105 dengan data berturut-turut 04, 01, dan 02. Untuk melakukan hal itu, tulislah perintah- perintah berikut ini:

Mengubah isi alamat $101 dengan data 04:
Ketik **MM 101** diikuti dengan tekan ENTER, akan muncul:
0101 05
Selanjutnya isilah dengan data 04 sehingga tampilannya:
0101 05 04 tekan ENTER.
Dengan cara tersebut anda telah mengubah isi alamat $101 dengan data $4. Lakukan hal yang sama untuk alamat $103 dengan data 01, dan alamat $105 dengan data 02.
Untuk memastikan bahwa anda telah berhasil melakukan perubahan isi memori lihat isi memori mulai alamat $100 dengan perintah MD 100 120 dan tekan ENTER. Pastikan bahwa hasil debugging program anda sesuai tampilan berikut ini.



Gambar 36. Tampilan Hasil Debugging Program

Perlu diperhatikan bahwa perbaikan program dengan cara seperti ini hanya akan mengubah kode-kode mesin yang ada pada RAM sistem 68HC11, tetapi tidak mengubah isi file .ASM, apalagi file .S19 dan file .LST.

17. Pengujian hasil debugging.

    Untuk menguji program yang telah diperbaiki, lakukan dengan cara trace. Ikuti perintah-perintah berikut ini:

Gambar 37. Tampilan Pengujian Hasil Debugging dengan Cara Trace

Catat keadaan register terutama isi PC dan akumulator A pada setiap baris program yang telah dilaksanakan! Catat pula isi alamat memori terutama lokasi penyimpan hasil pada kahir pelkasanaan seluruh baris program!

G. Tugas Akhir

1. Apa perbedaan simulator Wookie dengan program AxIDE dan BUFFALO?
2. Dalam proses komunikasi antara komputer host dengan sistem 68HC11, jelaskan fungsi program AxIDE dan BUFFALO!
3. Keuntungan apa saja yang bisa diperoleh oleh pemrogram dengan adanya fasilitas komunikasi antara komputer host dengan sistem 68HC11?
4. Apa perbedaan program .ASM, .S19, dan .LST?
5. Perhatikan isi file program .ASM berikut ini:

```
ORG     $0130
LDAA    #$0A
SUBA    #$09
ADDA    #$0B
SUBA    #$02
STAA    $0160
```

serta file .LST adalah:

```
0001 0130                ORG     $0130
0002 0130 86 0a          LDAA    #$0A
0003 0132 80 09          SUBA    #$09
0004 0134 8b 0b          ADDA    #$0B
0005 0136 80 02          SUBA    #$02
0006 0138 b7 01 60       STAA    $0160
```

Jika kode-kode mesin .S19 di atas anda download dari komputer host ke memori sistem 68HC11,

a. Gambarkan peta memori yang menunjukkan lokasi-lokasi dari kode-kode mesin file .S19 itu yang anda diperoleh dengan perintah MD 130 160!
b. Jelaskan makna kode-kode mesin itu dalam konteks format assembly!
c. Jika dilakukan trace sebanyak 5 kali, catat isi register PC, ACCA, dan isi alamat memori $0160 setiap suatu trace dilakukan! Jadi, berapa hasil akhir eksekusi program anda!

6. Pada praktikum pengujian hasil debugging, berdasarkan pengamatan terhadap register-register 68HC11 pada setiap baris program yang dieksekusi dan isi alamat memori pada akhir eksekusi program, apakah anda dapat menyimpulkan bahwa debugging anda telah berhasil dengan baik? Kemukakan alasan-alasannya!

BAB 4
PERCOBAAN III

Mode Pengalamatan Sistem 68HC11

A. Kompetensi Dasar

Mahasiswa mampu memahami mode pengalamatan M68HC11

B. Indikator Pencapaian Kompetensi

1. Menulis instruksi assembly dengan mode pengalamatan immediate, direct, extended, indexed, inherent, dan relative untuk mengakses resources mikrokontroler 68HC11
2. Memilih mode pengalamatan yang tepat sesuai dengan kebutuhan pemrograman sistem 68HC11

C. Dasar Teori

Dalam suatu pemrograman mikrokontroler, hal pertama yang harus dipahami oleh seorang pemrogram adalah model CPU yang digunakan. Model CPU umumnya direpresentasikan oleh nama dan spesifikasi register internalnya. Spesifikasi register internal mencakup panjang bit data yang dapat ditampung dan sifat register dalam menyimpan data seperti dengan cara paralel, serial, atau LIFO (last in-first out). Dengan kata lain, agar pemrogram mikrokontroler dapat mengerjakan program-programnya dengan benar dan mudah ia harus mengetahui terlebih dahulu nama dan spesifikasi register-register yang menjadi model CPU suatu mikrokontroler. Selain itu, hal penting lainnya yang harus diketahui oleh pemrogram adalah mode pengalamatan (addressing mode) yang berlaku pada mikrokontroler yang akan diprogram. Mode pengalamatan merupakan cara memberikan perintah untuk mengakses resources mikrokontroler seperti register, memori, dan port.

Pada sistem 68HC11 terdapat enam buah mode pengalamatan yang dapat digunakan untuk mengakses resources mikrokontroler yakni immediate, direct, extended, indexed, inherent, dan relative. Melalui percobaan ini anda akan belajar memberikan perintah menggunakan

berbagai mode pengalamatan yang berlaku pada mikrokontroler 68HC11.

D. Tugas Pendahuluan

1. Jelaskan model CPU untuk pemrograman mikrokontroler 68HC11 dengan menyebut nama dan spesifikasi register-registernya!
2. Jelaskan fungsi akumulator, register indeks, penunjuk tumpukan (stack ponter atau SP), program counter (PC), register kondisi atau register flag pada sistem 68HC11!
3. Apa yang dimaksud dengan opcode, mnemonic, dan operand dalam suatu instruksi assembly?
4. Sebutkan jenis-jenis mode pengalamatan pada mikrokontroler 68HC11 dan berikan contoh dua buah instruksi untuk masing-masing mode pengalamatan itu!
5. Tuliskan cara mengekspresikan bilangan 15 desimal, 1A2B heksadesimal, dan 01101111 biner dalam instruksi assembly 68HC11!
6. Tuliskan opcode dalam bentuk mnemonic perintah-perintah assembly untuk memuati akumulator A, memuati akumulator B, memuati akumulator D, memuati register indeks X, memuati register indeks Y, menjumlahkan isi akumulator A, menjumlahkan isi akumulator B, menjumlahkan isi akumulator D, mengurangkan isi akumulator A, mengurangkan isi akumulator B, mengurangkan isi akumulator D, mengirim isi akumulator A ke suatu alamat memori, mengirim isi akumulator B ke suatu alamat memori, mengirim isi akumulator D ke suatu alamat memeori, mengirim isi register indeks X ke suatu alamat memori, mengirim isi register indeks Y ke suatu alamat memori.
7. Apa arti opcode dalam bentuk mnemonic berikut ini: INCA, INCB DECA, DECB, INX, INY, JMP, BNE, dan BEQ.
8. Apa ciri-ciri instruksi dengan mode pengalamatan immediate dan instruksi dengan mode pengalamatan direct dari sisi format penulisan maupun tujuannya? Tulislah program assembly untuk meyimpan bilangan $7F ke alamat memori $50. Tunjukkan dengan memberikan keterangan di belakang instruksi-instruksi tersebut mana instruksi yang ditulis dengan mode pengalamatan immediate, dan mana yang ditulis dengan mode pengalamatan direct?

9. Tulislah program untuk membaca alamat $30, kemudian hasil pembacaan tersebut ditambahkan dengan bilangan $05, dan hasil akhirnya disimpan pada alamat memori $40. Anggap program dimulai pada alamat $20. Berikan keterangan di belakang baris program, baris-baris program itu termasuk jenis mode pengalamatan apa?
10. Jelaskan arti setiap baris program berikut ini dengan memberi keterangan di belakangnya tentang arti dan jenis mode pengalamatan yang digunakan:

```
LDAA    #$A2
STAA    $30
LDAB    #%10100111
STAB    $40
LDX     #$01
LDD     #15000
```

11. Perhatikan program berikut ini:

```
ORG     $0100
LDAA    $20
LDX     $30
```

Jika isi alamat memori diketahui seperti berikut ini:

```
>md 20 40

0020 AB FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0030 18 02 FF FF FF FF FF FF FF FF FF FF FF FF FF 08
0040 00 AB 18 FF AB CD 01 08 FF E4 E4 61 E3 C8 00 E4
```

Tentukan isi akumulator A, dan register indeks X dalam heksadesimal ketika seluruh baris program tersebut selesai dilaksanakan!

12. Program berikut ditulis dengan mode pengalamatan extended:

```
ORG     $0100
LDX     $0123
STX     $0130
```

Jika isi alamat $0100 sampai dengan $013F adalah:

```
0100 FE 01 23 FF 01 30 FF FF FF FF FF FF FF FF FF FF
0110 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0120 FF FF FF 25 12 FF FF FF FF FF FF FF FF FF FF FF
0130 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Jelaskan arti kode mesin pada alamat $0100 sampai dengan $0105! Jika program dijalankan, tulis isi alamat $0100 sampai dengan alamat $013F sebagai hasil eksekusi program!

13. Apa yang dimaksud dengan offset dan alamat efektif pada mode pengalamatan indexed dan berikan contohnya!
14. Perhatikan program yang ditulis dengan mode pengalamatan indexed berikut ini:

```
ORG     $2000
LDX     #$2000
LDAA    $10,X
ADDA    $20,X
SUBA    $30,X
ADDA    $40,X
STAA    $50,X
```

Berapa alamat efektif yang dibangkitkan oleh instruksi baris ke-3, ke-4, ke-5, ke-6, dan ke-7? Apa arti instruksi pada baris-baris tersebut?

15. Tulislah program pada tugas nomor 14 dengan mode pengalamatan extended!
16. Lakukan kompilasi program pada tugas nomor 14 dan program pada tugas nomor 15. Berdasarkan isi file .LST yang dihasilkan, mode pengalamatan apa yang lebih sedikit memerlukan alokasi memori? Jadi, apa keuntungan mode pengalamatan indexed dibandingkan dengan mode pengalamatan extended?
17. Apa ciri-ciri perintah mode pengalamatan inherent? Mengapa perintah- perintah dengan mode pengalamatan inherent tidak memerlukan operand?
18. Perhatikan program berikut ini:

```
ORG     $0100
LDAA    #$05
INCA
INCA
DECA
STAA    $0120
LDAB    #$03
INCB
STAB    $0130
LDX     #$1234
INX
STX     $0140
```

Berikan keterangan jenis mode pengalamatan di belakang setiap baris program! Jika program dijalankan secara keseluruhan, berapa isi alamat $0120, $0130, dan $0140?

19. Program berikut ini ditulis dengan mode pengalamatan relative:

```
        ORG     $0100       Alamat awal program $0100
        LDAA    #03         Memuati ACCA dengan 3 desimal
Awal    DECA                ACCA=ACCA-1 (decrement ACCA)
        STAA    $0120       Simpan isi ACCA ke alamat $0120
        BNE     AWAL        Ke AWAL jika ACCA ≠ 0
```

Jika program dijalankan, pada akhir eksekusi seluruh baris instruksi berapa isi alamat $0120? susun flowchart untuk program tersebut!

20. Jika hasil kompilasi program pada tugas 19 menghasilkan file .LST seperti berikut ini:

```
0001 0100                       ORG     $0100
0002 0100 86 03                 LDAA    #03
0003 0102 4a            AWAL    DECA
0004 0103 b7 01 20              STAA    $0120
0005 0106 26 fa                 BNE     AWAL
```

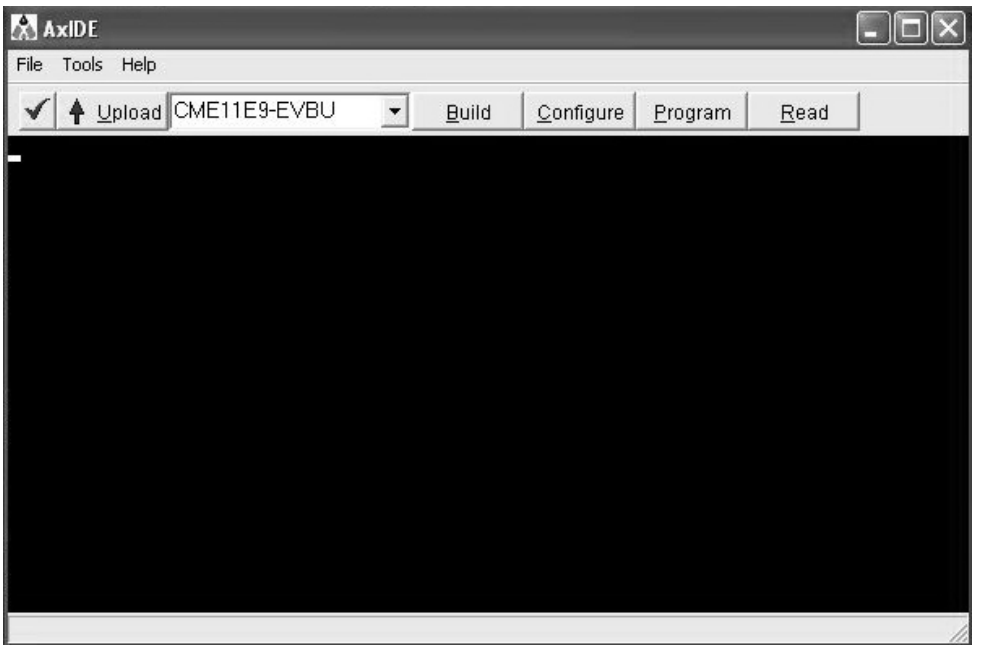
Berapa isi register PC jika kondisi percabangan pada baris perintah **BNE AWAL** benar yakni isi **ACCA ≠ 0**?

E. Alat-alat dan Langkah Percobaan

A. Satu unit sistem 68HC11 EVBU
B. Satu unit modul Basic I/O
C. Satu unit catu daya
D. Software AxIDE
E. Software Programmer's Notepad

F. Persiapan Percobaan

1. Pasang Kabel Serial Port COM pada Host PC.
2. Atur posisi jumper pada EVBU agar beroperasi pada mode single-chip dan Trace enable, yaitu dengan memasang jumper J3 dan J6.
3. Hubungkan Kabel Serial yang dengan EVBU.
4. Hubungkan Catu Daya dengan EVBU.
5. Aktifkan/Run program AxIDE.



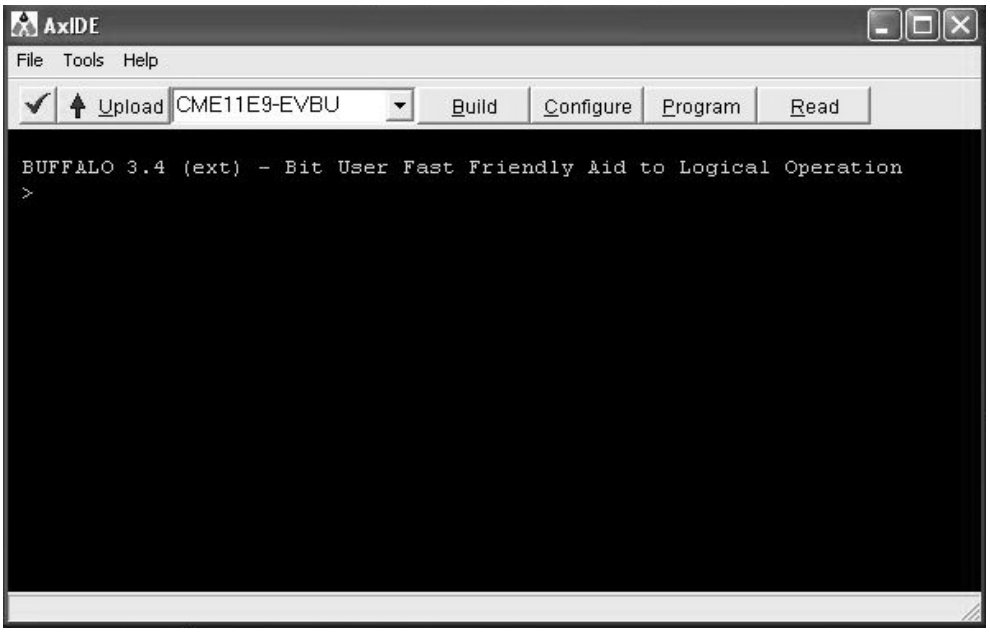
Gambar 38. Tampilan Program AxIDE

6. Click icon Check (Centang) atau melalui menu File>Option, pastikan setting komunikasi yang digunakan adalah sebagai berikut :

Tabel 6. Setting Komunikasi Program AxIDE

Port	COM1/COM2	Handshaking	Information
Baud Rate	9600		
Parity	None	Xon/Xoff	OFF
Data Bits	8	Rts/Cts	OFF
Stop Bits	1	Dtr/Dsr	OFF

   Jika belum sesuai lakukan pengaturan agar diperoleh setting seperti tampak pada tabel di atas.
7. Tekan tombol Reset pada EVBU.
8. Pastikan muncul tampilan BUFFALO pada terminal window AxIDE, yang menunjukkan bahwa EVBU dan komputer telah terhubung melalui Port Serial.
9. Tekan Enter agar anda berada pada prompt BUFFALO sebagai berikut:



Gambar 39. Tampilan Prompt BUFFALO Program AxIDE

Keterangan lebih lengkap ada pada petunjuk penggunaan software halaman 01 s/d 08.

10. Mode Pengalamatan Direct (1)
    Tulis program berikut ini dan simpan dengan nama DIRECT.ASM:

```
ORG     $20
LDAA    $30
LDX     $35
```

Lakukan assembly dan kompilasi dengan AxIDE terhadap program tersebut sehingga menghasilkan file DIRECT.S19 dan DIRECT.LST. Matikan catu daya sistem 68HC11, kemudian hidupkan kembali. Lakukan downloading file DIRECT.S19 dari komputer host ke memori sistem 68HC11. Lihat isi memori alamat $20 sampai dengan $30, pastikan isinya sebagai berikut:

```
>MD 20 30

0020 96 30 DE 35 FF FF FF FF FF FF FF FF FF FF FF FF
0030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Perhatikan bahwa kode-kode mesin dari program anda telah tersimpan pada alamat $20 sampai dengan $23! **Catat isi alamat $20 sampai dengan $23 tersebut!**

Ubahlah isi alamat memori **$30** menjadi **$02**, dan isi alamat **$35** menjadi **$ABCD**, dan setelah itu lihat isi alamat $20 sampai dengan $30!

```
>MM 30

0030 FF 02

>MM 35

0035 FF AB

>MM 36

0036 FF CD

>MD 20 30

0020 96 30 DE 35 FF FF FF FF FF FF FF FF FF FF FF FF
0030 02 FF FF FF FF AB CD FF FF FF FF FF FF FF FF FF
```

Catat isi alamat $30, $35, dan $36! Dengan menggunakan perintah **RM** ubahlah isi PC menjadi $20. Lakukan **trace** dua kali sehingga menghasilkan:

```
>RM
P-FFFF Y-FFFF X-FFFF A-FF B-FF C-D0 S-0047
P-FFFF 20

>T 2
LDAA $30              P-0022 Y-FFFF X-FFFF A-02 B-FF C-90 S-0047
LDX  $35              P-0024 Y-FFFF X-ABCD A-02 B-FF C-98 S-0047
```

Catat isi ACCA dan IX sebagai hasil pelaksanaan program!

11. Mode Pengalamatan Direct (2)

Ubahlah program pada file DIRECT.ASM menjadi sebagai berikut:

```
ORG     $20     Alamat awal program $20
LDAA    $30     Memuati ACCA dengan isi alamat $30
STAA    $80     Menyimpan isi ACCA ke alamat $80
LDX     $35     Memuati IX dengan isi alamat $35
STX     $90     Menyimpan isi IX ke alamat $90
```

Lakukan kompilasi terhadap program tersebut, RESET sistem 68HC11, kemudian lakukan downloading file DIRECT.S19 dari komputer host ke memori sistem 68HC11. Pastikan anda memiliki kode-kode mesin seperti berikut ini:

```
>load t
S10B002096309780DE35DF9075
S9030000FC

done
>MD 20 90

0020 96 30 97 80 DE 35 DF 90 FF FF FF FF FF FF FF FF
0030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0050 FF FF E4 E4 61 E3 C8 00 E4 61 E3 C8 00 5D 0D 20
0060 E4 F6 E8 00 E1 9A FF FF FF FF FF FF FF FF D0 00
0070 47 4D 44 20 32 30 20 39 30 0D FF FF FF FF FF FF
0080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0090 FF FF FF FF 4D 44 20 44 20 FF FF FF 00 90 00 00
```

Dengan perintah BUFFALO ubahlah isi alamat $30 menjadi $02 dan isi alamat $35 sampai dengan $36 menjadi $ABCD. Selanjutnya lihat isi memori 68HC11. Jika langkah anda benar akan menghasilkan tampilan:

```
>MM 30

0030 FF 02

>MM 35

0035 FF AB

>MM 36

0036 FF CD

>MD 20 90

0020 96 30 97 80 DE 35 DF 90 FF FF FF FF FF FF FF FF
0030 02 FF FF FF FF AB CD FF FF FF FF FF FF FF FF FF
0040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0050 FF FF E4 E4 61 E3 C8 00 E4 61 E3 C8 00 5D 0D 20
0060 E4 F6 E8 00 E1 9A FF FF FF FF FF FF FF FF D0 00
0070 47 4D 44 20 32 30 20 39 30 0D FF FF FF FF FF FF
0080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0090 FF FF FF FF 4D 44 20 44 20 FF FF FF 00 90 00 00
```

Catat isi alamat $30, $35, $36, $80, $90, dan $91 sebelum program dijalankan!

Ubah isi register PC menjadi PC=$20 kemudian lakukan trace empat kali, dan tampilkan isi alamat $20 sampai dengan $90. Lakukan langkah-langkah tersebut seperti tampilan berikut ini:

12. Mode Pengalamatan Extended

    Tulis program berikut ini dan beri nama EXTENDED.ASM:

ORG	$0100	Alamat awal program adalah $0100
LDAA	$0110	Memuati ACCA dengan isi alamat $110
STAA	$0120	Menyimpan isi ACCA ke alamat $0120
LDX	$0130	Memuati IX dengan isi alamat $0130
STX	$0140	Mengirim isi IX ke alamat $0140

Lakukan kompilasi, RESET sistem 68HC11, dan lakukan downloading file EXTENDED.S19 dari komputer host ke memori sistem 68HC11. Berikan perintah BUFFALO untuk melihat isi memori $0100 sampai dengan $0140! Jika langkah anda benar, akan diperoleh tampilan:

```
>LOAD T
S10F0100B60110B70120FE0130FF0140E1
S9030000FC

done
>MD 100 140

0100 B6 01 10 B7 01 20 FE 01 30 FF 01 40 FF FF FF FF
0110 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0120 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0130 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0140 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Perhatikan bahwa kode mesin program anda disimpan pada alamat memori $0100 sampai dengan $010B. Catat kode mesin pada alamat-alamat tersebut!

Ubah isi alamat memori $0110 dengan $12, dan alamat $0130 sampai dengan $0131 dengan $ABCD sehingga anda memperoleh tampilan berikut ini:

```
>MM 110

0110 FF 12

>MM 130

0130 FF AB

>MM 131

0131 FF CD

>MD 100 140

0100 B6 01 10 B7 01 20 FE 01 30 FF 01 40 FF FF FF FF
0110 12 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0120 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0130 AB CD FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0140 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Catat isi alamat $0110 dan alamat $0120. Catat pula alamat $0130 s.d. $0131 dan $0140 s.d. $0141 sebelum program dijalankan!

Ubahlah isi PC menjadi $0100 sesuai dengan alamat awal program anda! Jalankan program tersebut baris demi baris dengan memberikan perintah trace empat kali (T 4)! Lihat isi alamat

memori $0100 sampai dengan $0140! Jika langkah-langkah anda benar maka akan diperoleh tampilan layar PROCOMM seperti berikut ini:

```
>RM
P-FFFF Y-FFFF X-FFFF A-FF B-FF C-D0 S-0047
P-FFFF 100

>T 4
LDAA $0110            P-0103 Y-FFFF X-FFFF A-12 B-FF C-90 S-0047
STAA $0120            P-0106 Y-FFFF X-FFFF A-12 B-FF C-90 S-0047
LDX  $0130            P-0109 Y-FFFF X-ABCD A-12 B-FF C-98 S-0047
STX  $0140            P-010C Y-FFFF X-ABCD A-12 B-FF C-98 S-0047
>MD 100 140

0100 B6 01 10 B7 01 20 FE 01 30 FF 01 40 FF FF FF FF          0  @
0110 12 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0120 12 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0130 AB CD FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0140 AB CD FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Catat isi akumulator A, register indeks IX, dan alamat $0110, $0120, $0130 s.d. $0131, serta $0140 s.d. $0141 setelah program tersebut selesai dijalankan!

13. Mode Pengalamatan Indexed

Pada percobaan ini anda akan menggunakan sistem 68HC11 pada mode **expanded**! Lepaskan jumper **J4**! Sekarang anda telah bekerja dengan sistem 68HC11 mode expanded. Pada mode ini, memori yang digunakan adalah RAM eksternal yang dapat diakses mulai alamat $2000 (ingat rangkaian dekoder alamatnya!). Tulis program berikut ini dengan nama INDEXED.ASM:

```
ORG     $2000     Alamat awal program adalah $2000
LDX     #$2000    Memuati IX dengan bilangan $2000
LDAA    $10,X     Memuati ACCA dengan isi alamat $2010
ADDA    $20,X     Menambahkan isi ACCA dengan isi $2020
SUBA    $30,X     Mengurangkan isi ACCA dengan isi $2030
ADDA    $40,X     Menambahkan isi ACCA dengan isi $2040
STAA    $50,X     Menyimpan isi ACCA ke alamat $2050
```

Lakukan kompilasi, RESET sistem 68HC11, lakukan downloading file INDEXED.S19 dari host ke sistem 68HC11. Tampilkan isi alamat memori 2000 sampai dengan 2050. Pastikan bahwa anda memperoleh tampilan seperti di bawah ini.

```
>LOAD T
S1102000CE2000A610AB20A030AB40A750AE
S9030000FC

done
>MD 2000 2050

2000 CE 20 00 A6 10 AB 20 A0 30 AB 40 A7 50 21 4A 04
2010 00 00 68 00 01 16 00 20 00 01 00 01 01 00 00 00
2020 00 23 2D 01 90 CC 44 00 A6 0E 24 CA C0 FC 2D 80
2030 81 00 20 80 00 8C 00 00 45 00 B0 00 00 92 00 00
2040 06 00 80 EA 4F A2 E8 10 12 87 00 E4 F7 87 55 C2
2050 00 04 10 10 00 0A 02 00 20 00 00 0A 00 40 10 00
```

Catat isi alamat $2000 sampai dengan $200C. Buka file INDEXED.LST dengan Q editor

```
0001 2000                              ORG             $2000
0002 2000 ce 20 00                     LDX             #$2000
0003 2003 a6 10                        LDAA            $10,X
0004 2005 ab 20                        ADDA            $20,X
0005 2007 a0 30                        SUBA            $30,X
0006 2009 ab 40                        ADDA            $40,X
0007 200b a7 50                        STAA            $50,X
```

Hitung berapa byte kode-kode mesin anda menempati memori? Ubah isi alamat memori $2010 dengan $01, $2020 dengan $02, $2030 dengan $01, dan $2040 dengan $02, dan lihat isi alamat $2000 sampai dengan $2050 sehingga diperoleh tampilan sebagai berikut:

```
>md 2000 2050

2000 CE 20 00 A6 10 AB 20 A0 30 AB 40 A7 50 B9 4F 42
2010 01 00 20 00 01 12 00 00 00 01 00 00 00 00 00 00
2020 02 23 2D 09 D0 CC 6C 00 A6 5A 24 CA C0 FC AD 88
2030 01 00 00 80 00 88 00 00 04 00 B0 00 00 82 00 00
2040 02 00 80 EA 5F A2 E8 50 12 87 08 E4 F7 87 55 C3
2050 00 00 00 00 00 00 02 00 00 00 00 02 00 00 00 00
```

Catat isi alamat $2010, $2020, $2030, $2040, dan $2050 sebelum program dijalankan. Ubah nilai PC menjadi $2000 kemudian berikan perintah trace enam kali untuk menjalankan program baris demi baris. Setelah program selesai dijalankan seluruhnya Lihat isi alamat $2010, $2020, $2030, $2040, dan $2050, dan catatlah isinya. Catat pula isi register IX dan ACCA setelah semua baris program selesai dijalankan. Jika anda melakukan langkah-langkah itu dengan benar maka akan diperoleh tampilan seperti berikut ini:

```
>RM
P-200A Y-FFFF X-FFFF A-0B B-FF C-91 S-0047
P-200A 2000

>T 6
LDX  #$2000          P-2003 Y-FFFF X-2000 A-0B B-FF C-91 S-0047
LDAA $10,X           P-2005 Y-FFFF X-2000 A-01 B-FF C-91 S-0047
ADDA $20,X           P-2007 Y-FFFF X-2000 A-03 B-FF C-90 S-0047
SUBA $30,X           P-2009 Y-FFFF X-2000 A-02 B-FF C-90 S-0047
ADDA $40,X           P-200B Y-FFFF X-2000 A-04 B-FF C-90 S-0047
STAA $50,X           P-200D Y-FFFF X-2000 A-04 B-FF C-90 S-0047
>MD 2000 2050

2000 CE 20 00 A6 10 AB 20 A0 30 AB 40 A7 50 B9 4F 42          0 @ P OB
2010 01 00 20 00 01 12 00 00 00 01 00 00 00 00 00 00
2020 02 23 2D 09 D0 CC 6C 00 A6 5A 24 CA C0 FC AD 88  #-    l  Z$
2030 01 00 00 80 00 88 00 00 04 00 B0 00 00 82 00 00
2040 02 00 80 EA 5F A2 E8 50 12 87 08 E4 F7 87 55 C3      _  P       U
2050 04 00 00 00 00 00 02 00 00 00 00 02 00 00 00 00
```

Untuk menunjukkan perbedaan mode pengalamatan indexed dengan extended dari sisi alokasi memori yang digunakan, coba anda tulis program di atas yang menggunakan mode pengalamatan indexed yakni:

```
ORG     $2000    Alamat awal program adalah $2000
LDX     #$2000   Memuati IX dengan bilangan $2000
LDAA    $10,X    Memuati ACCA dengan isi alamat $2010
ADDA    $20,X    Menambahkan isi ACCA dengan isi $2020
SUBA    $30,X    Mengurangkan isi ACCA dengan isi $2030
ADDA    $40,X    Menambahkan isi ACCA dengan isi $2040
STAA    $50,X    Menyimpan isi ACCA ke alamat $2050
```

Menjadi program dengan menggunakan mode pengalamatan extended sebagai berikut:

```
ORG     $2000    Alamat awal program adalah $2000
LDAA    $2010    Memuati ACCA dengan isi alamat $2010
ADDA    $2020    Menambahkan isi ACCA dengan isi $2020
SUBA    $2030    Mengurangkan isi ACCA dengan isi $2030
ADDA    $2040    Menambahkan isi ACCA dengan isi $2040
STAA    $2050    Menyimpan isi ACCA ke alamat $2050
```

Kedua program tersebut akan memberikan hasil yang sama jika dijalankan. Hal yang membedakan adalah alokasi memori untuk kode-kode mesinnya! Lebih efisien yang mana dalam menggunakan memori? Untuk mengetahuinya, coba anda simpan program yang terakhir ke dalam file EXTEND2.ASM, lakukan kompilasi menjadi EXTEND2.S19, download EXTEND2.S19 dari komputer host ke sistem 68HC11. Lihat isi

memori mulai alamat $2000. Hitung berapa byte kode- kode mesin menempati lokasi memori!

Ubah isi alamat memori $2010 dengan $01, $2020 dengan $02, $2030 dengan $01, dan $2040 dengan $02, sama seperti pada mode pengalamatan indexed dan lihat serta catat isi alamat $2050. Ubah isi PC menjadi $2000 kemudian trace 5 kali, lihat dan catat isi alamat $2050!

14. Mode Pengalamatan Inherent

Pasang jumper pada J4 agar sistem beroperasi pada mode single chip dan pasang jumper pada J7 agar program monitor BUFFALO dapat digunakan untuk debugging dengan baik terutama untuk trace.

Tulis program assembly berikut ini:

```
ORG     $0100
LDAA    #$05
INCA
INCA
DECA
LDAB    #$03
INCB
LDX     #$03
INX
```

Simpan program ke dalam file INHERENT.ASM, lakukan kompilasi menjadi file INHERENT.S19, downloading file INHERENT.S19 dari komputer host ke sistem 68HC11. Lihat dan catat isi alamat $100 sampai dengan $10B!

Ubah isi register PC menjadi $100, karena program anda beralamat awal $0100, kemudian lakukan trace delapan kali! Catat isi ACCA, ACCB, dan IX setelah program selesai dijalankan!

15. Mode Pengalamatan Relative

Tulis program assembly berikut ini:

```
      ORG    $0100     Alamat awal program $0100
      LDAA   #03       Memuati ACCA dengan 3 desimal
Awal  DECA             ACCA=ACCA-1 (decrement ACCA)
      STAA   $0120     Simpan isi ACCA ke alamat $0120
      BNE    AWAL      Ke AWAL jika ACCA ≠ 0
```

Simpan program tersebut dalam file RELATIVE.ASM, kompilasi menjadi RELATIVE.S19, dan lakukan downloading kode-kode

mesin tersebut ke sistem 68HC11! Lihat isi alamat memori $100 sampai dengan $107.
Gunakan perintah RM untuk mengatur agar PC=$0100. Lakukan trace 10 kali! Catat isi PC dan ACCA setiap suatu baris instruksi selesai dijalankan!

G. Tugas Akhir

1. Berdasarkan hasil percobaan anda pada prosedur nomor 6 yakni mode pengalamatan immediate:
   a. Berapa isi akumulator A (ACCA) pada akhir trace ke-2, dan apakah isi ACCA itu telah sesuai dengan perintah yang diberikan yakni **LDAA #$03** dan **SUBA #$01**?
   b. Berapa isi akumulator B (ACCB) pada akhir trace ke-4, dan apakah isi ACCB itu telah sesuai dengan perintah yang diberikan yakni **LDAB #%10100111**?
   c. Pada akhir trace ke-5 berapa isi register indeks IX, dan apakah isinya sesuai dengan perintah **LDX #$1234**?
   d. Pada akhir trace ke-6 berapa isi akumulator D, dan apakah isinya sesuai dengan perintah **LDD #15000**?
   e. Tunjukkan bahwa suatu operand dengan sistem bilangan biner, desimal, dan heksadesimal akan selalu disimpan ke register dalam sistem heksadesimal!
   f. Apakah dengan mode pengalamatan immediate anda dapat memindahkan data ke alamat memori? Kalau begitu apa fungsi suatu perintah dengan mode pengalamatan immediate, dan mode itu mentransfer data dari sumber berbentuk apa ke tujuan transfer berbentuk apa?
2. Dari hasil percobaan anda tentang mode pengalamatan direct bagian 1 yakni prosedur butir 7:
   a. Tunjukkan mana opcode dan operand low order-byte pada kode-kode mesin yang tersimpan pada alamat $20 sampai dengan $23! Berapa operand high-order byte dari mode pengalamatan direct ini, dan apakah operand itu dimasukkan pada memori? Jadi, apa keuntungan mode pengalamatan ini?
   b. Berapa isi ACCA, bandingkan dengan isi alamat memori $30 dan apakah isinya sesuai dengan perintah **LDAA $30**?
   c. Berapa isi register IX, bandingkan dengan isi alamat memori $35 sampai dengan $36, dan apakah isinya sesuai dengan perintah **LDAA $35**?

d. Apakah anda melihat bahwa perintah-perintah itu akan memindahkan data dari alamat memori ke register? Kalau begitu kemukakan salah satu fungsi perintah dengan mode pengalamatan direct!

3. Berdasarkan percobaan tentang mode pengalamatan direct bagian ke-2 seperti pada prosedur butir 8:
   a. Sebelum program dijalankan, berapa isi alamat $30, dan $80, apakah isinya sama?
   b. Setelah program dijalankan, berapa isi alamat $30 dan $80, apakah isinya sama? Apa artinya? Perintah mana yang menyebabkan hal itu terjadi?
   c. Sebelum program dijalankan, berapa isi alamat $35 s.d. $36 dan $90 s.d. $91, apakah isinya sama?
   d. Setelah program dijalankan, berapa isi alamat $35 s.d. $36 dan $90 s.d. $91,apakah isinya sama? Apa artinya? Perintah mana yang menyebabkan hal itu terjadi?
   e. Apakah anda melihat bahwa perintah-perintah itu akan memindahkan isi suatu alamat memori ke alamat memori yang lain? Jika demikian, kecuali berfungsi memindah data dari sumber berbentuk alamat memori ke ke tujuan berbentuk register, mode pengalamatan direct akan memindahkan data dari sumber berbentuk apa ke tujuan berbentuk apa? Berikan penjelasan dengan kalimat yang tepat!
4. Atas dasar prosedur percobaan butir 9 tentang mode pengalamatan extended:
   a. Jelaskan arti kode mesin yang menempati alamat $0100 sampai dengan $010B! Tunjukkan pada mode pengalamatan extended, alamat efektif operand disimpan pada dua byte setelah opcode! Tunjukkan pula alamat- alamat memori yang menyimpan operand high-order byte dan operand low-order byte dari mode pengalamatan extended, dan dalam hal ini apa perbedaannya dengan mode pengalamatan direct? Berapa isi akumulator A dan register indeks IX setelah program dijalankan? Apakah sudah sesuai dengan perintah yang diberikan? Tunjukkan perintah-perintah mana yang menyebabkan pengisian akumulator A dan register indeks IX?
   b. Berapa isi alamat memori $0110 dan $0120 sebelum program dijalankan? Apakah sama? Setelah program dijalankan, berapa isi kedua alamat memori tersebut? Samakah isinya?

c. Berapa isi alamat $0130 s.d. $0131 dan alamat $0140 s.d. $0141 sebelum program dijalankan, dan apakah isi keduanya sama? Setelah program dijalankan berapa isi keduanya, dan apakah sama? Apakah anda melihat bahwa perintah pada mode pengalamatan extended memiliki fungsi yang sama dengan mode pengalamatan direct?
d. Kemukakan apa fungsi perintah dengan mode pengalamatan extended? Jelaskan perbedaan dan persamaannya dengan perintah dengan mode pengalamatan direct!

5. Dari percobaan mode pengalamatan indexed pada butir 10 porosedur percobaan:
   a. Bandingkan isi alamat memori $2000 sampai dengan $200C dengan isi alamat yang sama pada file INDEXED.LST! Apakah sama? Jadi, berapa byte kode-kode mesin anda ditempatkan di dalam memori?
   b. Perhatikan isi alamat memori $2010, $2020, $2030, $2040, dan $2050 sebelum maupun setelah program dijalankan, isi register IX dan ACCA untuk setiap baris program yang dijalankan dan program assembly yang telah anda tulis! Apakah program telah dapat berjalan dengan baik? Tunjukkan bahwa setiap baris program telah dapat membangkitkan alamat efektif sesuai dengan offset yang diberikan?
   c. Bandingkan isi alamat $2050 setelah running program antara mode pengalamatan indexed dan mode pengalamatan extended! Apakah hasilnya sama? Apakah dapat disimpulkan bahwa kedua progam itu sama? Berapa byte kode-kode mesin program yang ditulis dengan mode pengalamatan extended? Bandingkan dengan jumlah lokasi yang ditempati kode-kode mesin untuk program yang ditulis dengan mode pengalamatan indexed! Jadi, lebih efisien mana dalam hal penempatan kode-kode mesin di memori untuk kedua jenis pengalamatan tersebut?
6. Berdasarkan prosedur percobaan butir 11 tentang mode pengalamatan inherent:
   a. Perhatikan isi alamat $100 sampai dengan $10B! Tunjukkan bahwa pada mode pengalamatan inherent, instruksinya hanya menempati 1 byte alamat memori!
   b. Perhatikan isi register ACCA, ACCB, dan IX setelah program dijalankan! Jadi, apa arti INCA, DECA, INCB, dan INX!
7. Dari percobaan mode pengalamatan relative pada butir 12:

a. Pada alamat berapa perintah dengan label AWAL yakni DECA ditempatkan?
b. Jika pada perintah pengalamatan relative **BNE AWAL** kondisinya terpenuhi yakni ACCA≠0, berapa isi PC? Jadi, program melompat ke alamat berapa?
c. Terdapat berapa loop pada program tersebut? Kapan program keluar dari loop?

BAB 5
PERCOBAAN IV

Pemrograman Port Input/Output pada EVBU M68HC11

Percobaan 4. A
Pemrograman Port Input/Output pada Mode Single Chip

A. Kompetensi Dasar

Mahasiswa mampu memahami Pemrograman Port Input/Output pada EVBU M68HC11 pada mode single chip

B. Indikator Pencapaian Kompetensi

Melalui percobaan ini mahasiswa diharapkan dapat memrogram port yang tersedia pada sistem M68HC11 sebagai saluran input dan output serbaguna

C. Dasar Teori

Sistem M68HC11 mempunyai lima buah port yang dapat digunakan sebagai port I/O paralel serbaguna maupun fungsi lain.

Tabel 7. Port Input/Output

Port	Pin input	Pin output	Pin dua arah	Fungsi lain
Port A	3	3	2	Timer
Port B	-	8	-	Alamat orde tinggi
Port C	-	-	8	Alamat orde rendah dan bus data
Port D	-	-	6	SCI dan SPI
Port E	8	-	-	Konverter A/D

Pengaksesan data pada port paralel dilakukan dengan operasi baca/tulis register internal untuk port tersebut. Fungsi sebuah port ditentukan oleh isi register kontrol untuk port tersebut. Register internal dan register kontrol terletak pada alamat $1000 hingga $10FF.

1. Port A

   Port A mempunyai tiga pin masukan, tiga pin keluaran, dan dua pin dua arah. Port A juga berfungsi sebagai Sistem Timer. Data pada port dapat diakses melalui register PORTA pada alamat

$1000. susunan bit bit register PORTA jika berfungsi sebagai port I/O adalah:

Bit 7	6	5	4	3	2	1	Bit 0
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0

Setelah reset PORTA akan berisi %10001111. PA0, PA1, dan PA2 adalah masukan. PA4, PA5, dan PA6 adalah keluaran. PA3 dan PA7 adalah dua arah. Susunan bit register PORTA bila berfungsi sebagai Sistem Timer adalah:

Bit 7	6	5	4	3	2	1	Bit 0
PAI	OC2	OC3	OC4	IC4/OC5	IC1	IC2	IC3

Atau

Bit 7	6	5	4	3	2	1	Bit 0
OC1	OC1	OC1	OC1	OC1	-	-	-

Arah data pada pin dua arah ditentukan oleh isi register kontrol yaitu PACTL. Register ini juga berfungsi sebagai register kontrol untuk Pulse Accumulator. Susunan bit untuk register ini adalah:

Bit 7	6	5	4	3	2	1	Bit 0
DDRA7	PAEN	PAMOD	PEDGE	DDRA3	I4/O5	RTR1	RTR0

Bit yang menentukan arah data pada PA3 dan POA7 adalah DDRA3 dan DDRA7. jika berisi ”0” maka masuk, bila ”1” maka keluar. Bit PAEN digunakan untuk mengaktifkan Pulse Accumulator. Jika PAEN berisi ”0” maka Pulse Accumulator tidak aktif dan port A berfungsi sebagai port I/O paralel. Jika PAEN berisi ”1” maka Pulse Accumulator aktif dan port A tidak dapat dipakai sebagai port I/O paralel.

2. Port B

Pada mode single-chip dan bootstrap, port B berfungsi sebagai port output serbaguna (general purpose output). Pada mode expanded dan special test port B berfungsi sebagai jalur keluar alamat orde tinggi (high order address). Dalam mode PROG, port B berfungsi sebagai jalur masuk alamat orde tinggi.

Data pada port dapat diakses melalui register PORTB dengan alamat $1004. susunan bit port B sebagai port output adalah:

Bit 7	6	5	4	3	2	1	Bit 0
PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0

Susunan PORTB sebagai jalur alamat orde tinggi adalah:

Bit 7	6	5	4	3	2	1	Bit 0
DDR15	DDR14	DDR13	DDR12	DDR11	DDR10	DDR9	DDR8

3. Port C
   Pada mode single-chip dan bootstrap, port C berfungsi sebagai port input/output (dua arah, bidirectional). Pada mode expanded dan special test, port C berfungsi sebagai jalur data yang dimultiplekskan dengan jalur alamat orde rendah. Data pada potr C dapat diakses melalui register PORTC. Susunan register PORTC bila berfungsi sebagai port input/output adalah:

Bit 7	6	5	4	3	2	1	Bit 0
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

Susunan bit register PORTC bila berfungsi sebagai jalur alamat orde rendah adalah:

Bit 7	6	5	4	3	2	1	Bit 0
ADDR7	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0

Arah data untuk masing-masing bit pada port C ditentukan oleh isi register DDRC dengan alamat $1007. Jika pada suatu bit berisi ”0” maka bit tersebut berfungsi sebagai input. Jika berisi ”1” maka bit tersebut berfungsi sebagai output. Susunan bit register DDRC adalah:

Bit 7	6	5	4	3	2	1	Bit 0
DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0

4. Port D
   Port D mempunyai 6 bit jalur data yang dapat digunakan sebagai input/output serba guna atau sebagai jalur SCI (Serial Communication Interface) dan SPI (Serial Peripheral Interface).

Data pada port dapat diakses melalui register PORTD dengan alamat $1008. Susunan bit register PORTD sebagai port input/output adalah:

Bit 7	6	5	4	3	2	1	Bit 0
0	0	PD5	PD4	PD3	PD2	PD1	PD0

Susunan bit register PORTD sebagai jalur SCI dan SPI adalah:

Bit 7	6	5	4	3	2	1	Bit 0
0	0	SS	SCK	MOSI	MISO	TxD	RxD

Setelah reset PORTD berisi %00111111. Arah data pada port D ditentukan oleh isi register DDRD pada alamat $1009. Susunan bit register DDRD adalah:

Bit 7	6	5	4	3	2	1	Bit 0
0	0	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0

Isi bit ”0” menandakan input dan ”1” menandakan output. Setelah reset DDRD berisi %00000000.

5. Port E

Port E adalah port input 8 bit dengan fungsi lain sebagai jalur masuk konverter analog ke digital. Data port dapat diakses melalui register PORTE dengan alamat $100A. Susunan register PORTE sebagai port input adalah:

Bit 7	6	5	4	3	2	1	Bit 0
PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0

Susunan register PORTE sebagai jalur masukan konverter A/D adalah:

Bit 7	6	5	4	3	2	1	Bit 0
AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0

Setelah reset PORTE berisi %11111111

Port E dapat difungsikan sebagian sebagai port input dan sebagian lagi sebagai jalur masuk konverter A/D dengan syarat

register PORTE jangan dibaca selama pencuplikan dalam siklus konversi A/D.

D. Tugas Pendahuluan

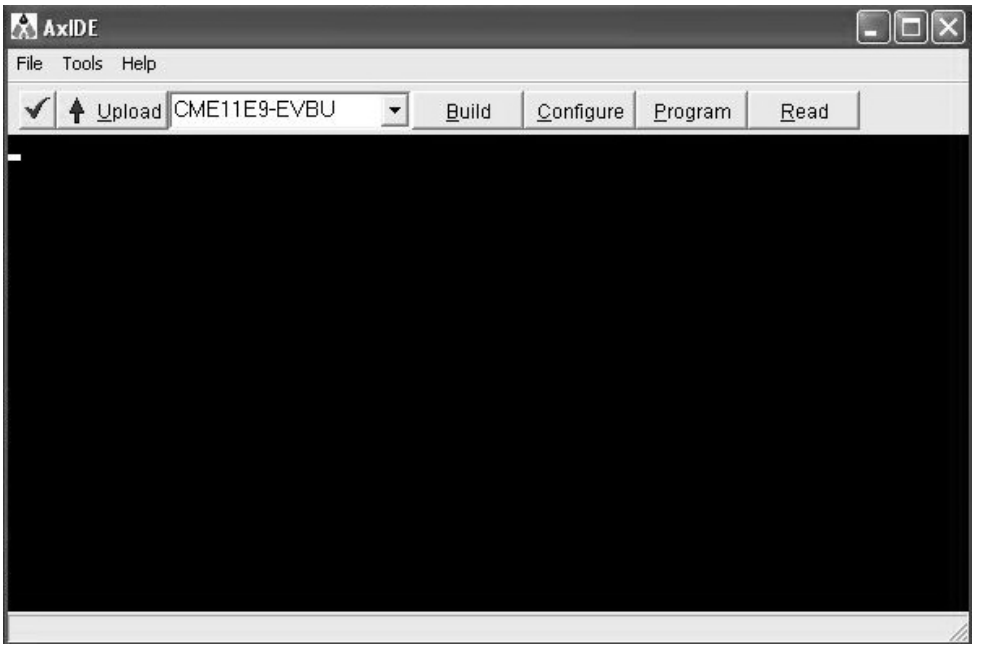
Apakah yang anda ketahui tentang mode operasi sigle chip? Resources M68HC11 apa sajakah yang dapat diakses pada mode ini?

E. Alat-alat dan Langkah Percobaan

1. Satu unit sistem 68HC11 EVBU
2. Satu unit modul Basic I/O
3. Satu unit catu daya
4. Software AxIDE
5. Software Programmer's Notepad

F. Persiapan Percobaan

1. Pasang Kabel Serial Port COM pada Host PC.
2. Atur posisi jumper pada EVBU agar beroperasi pada mode single-chip dan Trace enable, yaitu dengan memasang jumper J3 dan J6.
3. Hubungkan Kabel Serial yang dengan EVBU.
4. Hubungkan Catu Daya dengan EVBU.
5. Aktifkan/Run program AxIDE.



Gambar 40. Tampilan Program AxIDE

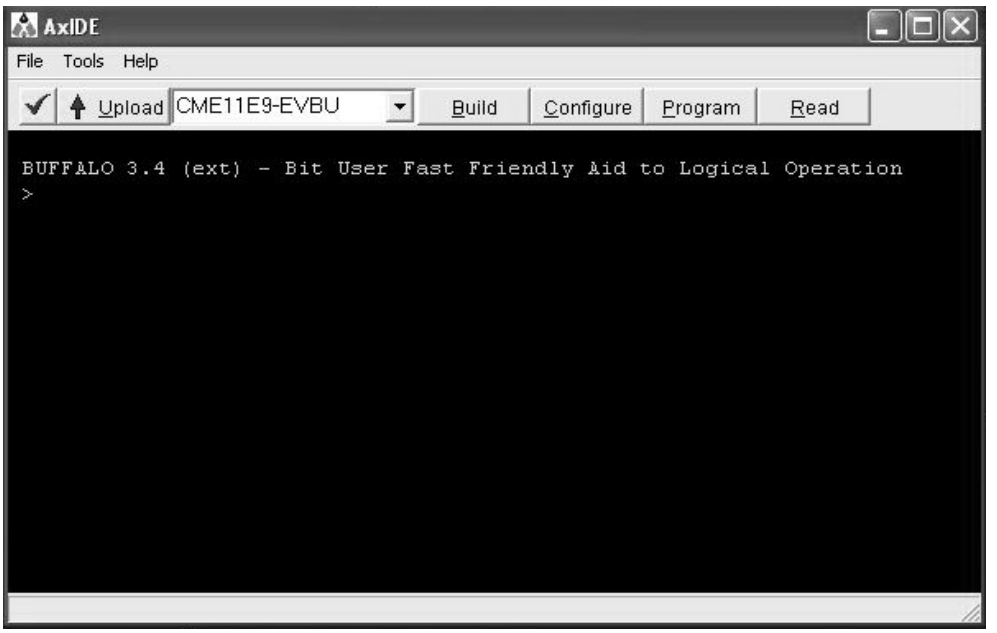
6. Click icon Check (Centang) atau melalui menu **File>Option**, pastikan setting komunikasi yang digunakan adalah sebagai berikut :

Tabel 8. Setting Komunikasi Program AxIDE

Port	COM1/COM2	Handshaking	Information
Baud Rate	9600		
Parity	None	Xon/Xoff	OFF
Data Bits	8	Rts/Cts	OFF
Stop Bits	1	Dtr/Dsr	OFF

Jika belum sesuai lakukan pengaturan agar diperoleh setting seperti tampak pada tabel di atas.

7. Tekan tombol Reset pada EVBU.
8. Pastikan muncul tampilan BUFFALO pada terminal window AxIDE, yang menunjukkan bahwa EVBU dan komputer telah terhubung melalui Port Serial.
9. Tekan Enter agar anda berada pada prompt BUFFALO sebagai berikut:



Gambar 41. Tampilan Prompt BUFFALO Program AxIDE

Keterangan lebih lengkap ada pada petunjuk penggunaan software halaman 01 s/d 08.

10. Mengirim Data ke Port B

Hubungkan kabel 60pin dari EVBU ke modul BASIC I/O. Dengan langkah ini berarti anda telah memasang 8 buah LED pada pin port B, dan 8 buah saklar pada pin port C. Program berikut ini akan menampilkan bilangan $A1 heksadesimal pada port B secara berkedip. Menggunakan software Programmer's Notepad, tulis program berikut ini:

```
PORTB   EQU     $1004       Penyamaan PORTB dengan alamat $1004
        ORG     $0100       Alamat awal program adalah $0100
AWAL    LDAA    #$A1        Mengisi ACCA dengan %00100111
        STAA    PORTB       Mengirim isi ACCA ke PORTB
        LDY     #$FFFF      Instruksi untuk memberikan tunda
LOOP1   DEY                         pada nyala LED sebesar $FFFF
        BNE     LOOP1
        LDAA    #$00        Mengisi ACCA dengan $00
        STAA    PORTB       Mengirim isi ACCA ke PORTB untuk
        LDY     #$FFFF      Instruksi untuk memberikan tunda
LOOP2   DEY                         pada LED padam sebesar $FFFF
        BNE     LOOP2
        JMP     AWAL        Lompat ke AWAL
```

Simpan dengan namam SBM1A1.asm. Lakukan assembly melalui program AxIDE, Download hasil assembly ke EVBU. Jalankan program dengan perintah GO 100. Amati LED yang terhubung ke port B! Apakah keadaan nyala LED sesuai dengan data %1010 0001? Coba data yang dikirim ke port B anda ganti dengan $27, $35, dan $4B!

11. Membaca Data Pada Port C dan Mengirim Hasilnya ke Port B

Program berikut ini digunakan untuk membaca PORTC, dan hasilnya dijumlahkan dengan bilangan heksadesimal $12 serta hasil akhirnya dikirim ke PORTB. Menggunakan Programmer's Notepad, tulis program berikut ini:

```
PORTB   EQU   $1004    Penyamaan PORTB dengan alamat $1004
PORTC   EQU   $1003    Penyamaan PORTC dengan alamat $1003
DDRC    EQU   $1007    Penyamaan DDRC dengan alamat $1007
        ORG   $0100    Alamat awal program adalah $0100
        LDAA  #$00     Memuati ACCA dengan %00000000
        STAA  DDRC     Mengatur PORTC sebagai input
AWAL    LDAA  PORTC    Membaca data pada PORTC
        ADDA  #$12     Menambahkan ACCA dengan $12
        STAA  PORTB    Mengirim data yang dibaca ke PORTB
        JMP   AWAL     Program melompat ke AWAL
```

Simpan dengan nama SBM1A2.asm. Lakukan assembly melalui program AxIDE, Download hasil assembly ke EVBU. Jalankan program dengan perintah GO 100. Ubahlah posisi saklar pada port C misalnya membentuk konfigurasi 10101010, amati LED pada port B! Apakah LED pada port B telah menyala sesuai dengan data yang dimasukkan lewat port C ditambah dengan $12? Ubah program agar bisa menampilkan data pada port B yang merupakan hasil pembacaan port C dikurangi $02!

12. Lampu Berjalan Pada Port B

Program berikut ini digunakan untuk mengatur nyala LED pada PORTB, yaitu nyala LED berjalan (bergeser) dari kanan ke kiri. Menggunakan Programmer's Notepad, tulis program berikut ini:

```
PORTB   EQU   $1004    Penyamaan PORTB dengan alamat $1004
        ORG   $0100    Alamat awal program adalah $0100
        LDAA  #$01     Memuati ACCA dengan %00000001
AWAL    STAA  PORTB    Mengirim data ke PORTB
        LDY   #$FFFF   Memuati IY dengan $FFFF (waktu tunda)
TUNDA   DEY            Decrement IY, IY=IY-1
        BNE   TUNDA    Lompat ke TUNDA jika IY≠0
        ROLA           Isi ACCA geser ke kiri 1-bit
        JMP   AWAL     Program melompat ke AWAL
```

Simpan dengan nama SBM1A3.asm. Jalankan program dengan perintah GO 100. Amati LED pada port B! Apakah nyala LED bergeser setiap saat sehingga nampak seperti lampu berjalan? Coba anda percepat bergesernya nyala LED dengan mengubah waktu tunda dari $FFFF menjadi $3FFF. Ubahlah program agar nyala LED berjalan ke kanan!

G. Tugas Akhir

1. Pada Laporan gambarkan flow-chart untuk program di atas.
2. Susun program agar port D dapat berfungsi sebagai saluran input. Selanjutnya, data hasil pembacaan pada port D ditambah 1 melalui operasi increment, dan hasilnya disimpan pada alamat $0170 serta dikirim ke port B.(lengkapi dengan flow chart pada laporan anda)
3. Tulis program untuk membaca data pada port E. Hasil pembacaan data pada port E disimpan pada alamat $0170 dan dikirim ke port D serta port B!
4. Susun program agar pin PA0, PA1, PA2, dan PA3 pada port A dapat difungsikan sebagai saluran input, pin PA4, PA5, PA6, dan PA7 sebagai saluran output. Selanjutnya data dari saluran input port A dibaca, dan hasilnya dikirim ke saluran output port A.

Percobaan 4. B

Pemrograman Port Input/Output pada Mode Expanded dengan Programmable Peripheral Interface (PPI 8255)

A. Kompetensi Dasar

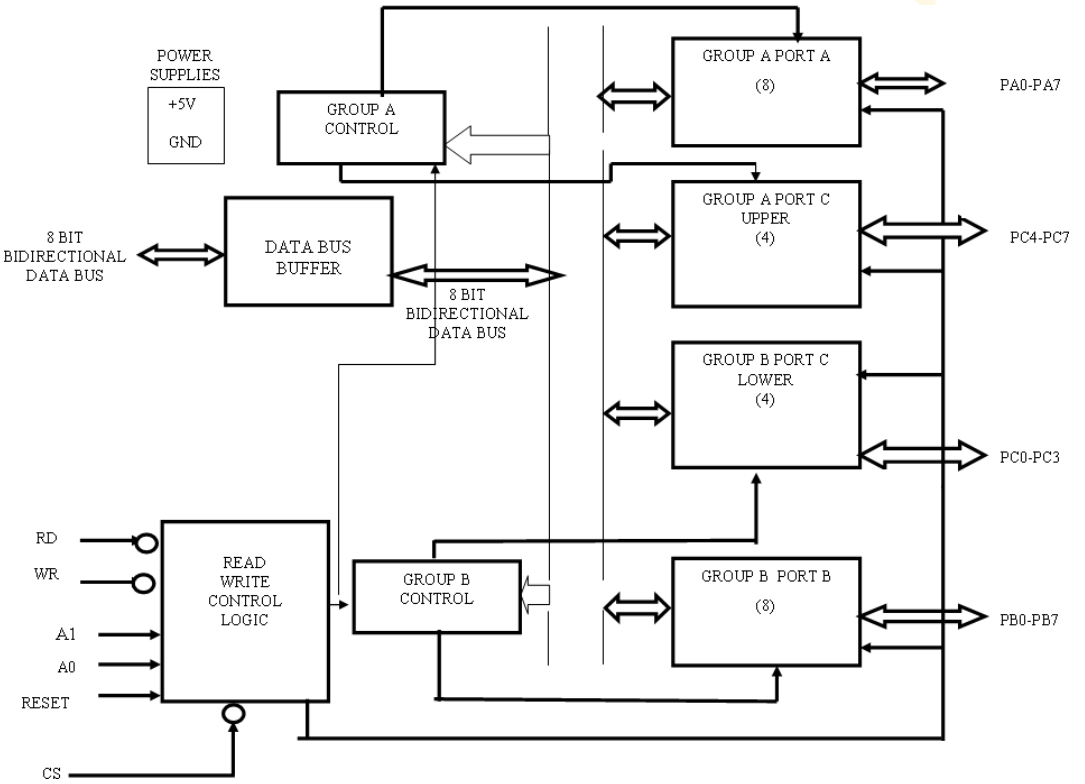
Mahasiswa mampu memahami mode operasi expanded M68HC11 berbasi PPI8255

B. Indikator Pencapaian Kompetensi

1. Menyusun rangkaian dekoder alamat PPI 8255 untuk diinterkoneksikan dengan sistem 68HC11
2. Memrogram PPI 8255 pada mode 0

C. Dasar Teori

PPI 8255 merupakan chip (IC) yang menyediakan fungsi antarmuka input/output antara peripheral dengan mikroprosesor yang dapat diprogram. Piranti ini disediakan oleh pabriknya dalam kemasan DIP (Dual In Line Package) memiliki pin (kaki) sebanyak 40 buah, terdiri 24 kaki untuk port I/O yakni port A (8-bit), port B (8-bit), dan port C (2 x 4-bit), 2 port untuk catudaya (Ground dan Vcc), 8-bit saluran data, 2-bit alamat, kaki kontrol WR, RD, CE, dan RESET. Diagram internal PPI 8255 tergambar pada gambar dibawah ini.



Gambar 42. Diagram Internal PPI 8255

Pada gambar di atas, buffer bus data 8 bit berfungsi sebagai penghubung dengan bus data sistem yang bersifat tristate dan bidirectional. Data yang dikirim dan diterima oleh PPI 8255 terdiri atas 4 bit yang di-latch dan dapat digunakan untuk mengontrol sinyal keluaran dan status sinyal masukan yang berhubungan dengan port A dan port B.

Tabel 9. Operasi Dasar PPI 8255

A1	A0	RD	WR	CS	OPERASI	ARAH DATA
0	0	0	1	0	CPU Baca	Port A ke Bus Data
0	1	0	1	0	CPU Baca	Port A ke Bus Data
1	0	0	1	0	CPU Baca	Port A ke Bus Data
0	0	1	0	0	CPU Tulis	Port Data ke Port A
0	1	1	0	0	CPU Tulis	Port Data ke Port A
1	0	1	0	0	CPU Tulis	Port Data ke Port A
1	1	1	0	0	CPU Tulis	Bus Data ke Reg. Control
X	X	X	X	1	CPU Tulis	Bus Data Tristate
1	1	1	1	0	CPU Tulis	Bus Data Tristate
1	1	0	1	0	CPU Tulis	Kondisi Illegal

1. Mode Operasi 8255
   PPI 8255 mempunyai 3 macam mode operasi yang dikendalikan melalui perngkat lunak ke masing-masing port, yaitu :
   a. Mode 0 (Masukan/Keluaran Dasar)
      Secara fisik chip PPI 8255 memiliki 40 pin yang fungsinya terbagi menjadi beberapa bagian seperti ditunjukkan pada tabel 2. Untuk mengatur fungsi semua port yang ada sebagai input atau output dilakukan pemrogaman yakni dengan cara mengisi register control word yang dimilikinya dengan suatu control word yang nilainya sesuai dengan fungsi yang diinginkan. Mode pemrograman PPI 8255 terbagi menjadi tiga yakni mode 0 atau basic I/O, mode 1 atau strobed, dan mode 2 atau bidirectional. Prinsip pemrograman PPI 8255 adalah mengirim control word yang diperlukan ke alamat register control word. Alamat register control word dan port yang ada dapat diakses dengan memberi sinyal alamat ke pin alamat PPI 8255 yakni A1 dan A0. Tabel 2 menunjukkan kombinasi sinyal alamat yang digunakan untuk mengakses register control word dan semua port yang disediakan oleh chip PPI 8255.

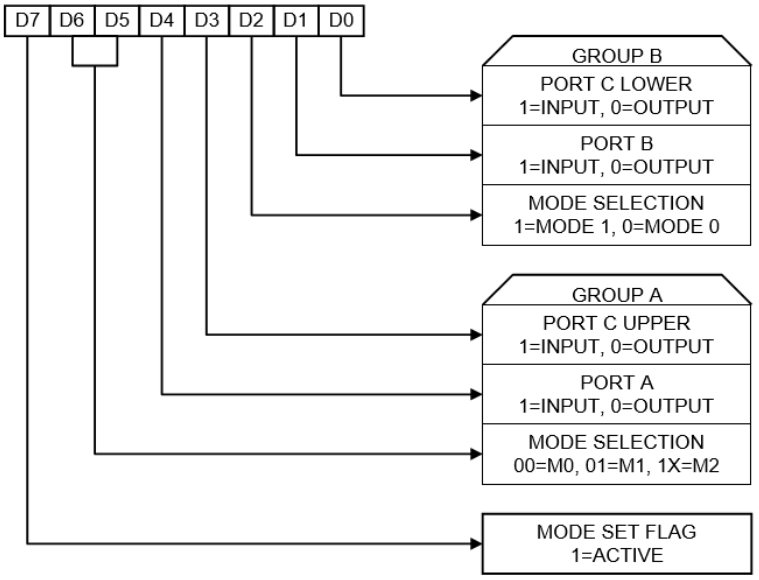
Tabel 10. Fungsi pin pada chip PPI 8255

Notasi Pin	Nama Pin	Interkoneksinya dengan mikroprosesor
D0 s.d. D7	Jalur Data 2-Arah 8-bit	Bus Data
A0 s.d. A1	Jalur Alamat 2-bit	Bus Alamat
PA0 s.d. PA7	Port A 8-bit	-
PB0 s.d. PB7	Port B 8-bit	-
PC0 s.d. PC3	Port C Lower 4-bit	-
PC4 s.d. PC7	Port C Upper 4-bit	-
RESET	Input RESET	RESET
$\overline{RD}$	Input Read (ACTIVE-LOW)	$\overline{IOR}$ atau $\overline{OE}$
$\overline{WR}$	Input Write (ACTIVE-LOW)	$\overline{IOW}$ atau $\overline{WE}$
$\overline{CE}$	Input ChipEnable (ACTIVE-LOW)	Bus Alamat Lewat Dekoder Alamat
+Vcc	Sumber Tegangan +	+Vcc
Ground	Sumber Tegangan 0	Ground

Tabel 11. Sinyal alamat untuk mengakses register control word dan semua port pada chip PPI 8255

Sinyal Alamat		Unit yang diakses
A1	A0	
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Register Control Word

Bedasarkan data sheet yang diterbitkan oleh pabriknya, cara menentukan nilai control word yang diperlukan untuk mengatur fungsi port pada chip PPI 8255 adalah dengan bantuan diagram berikut ini.



Gambar 43. Pengaturan control word pada PPI 8255

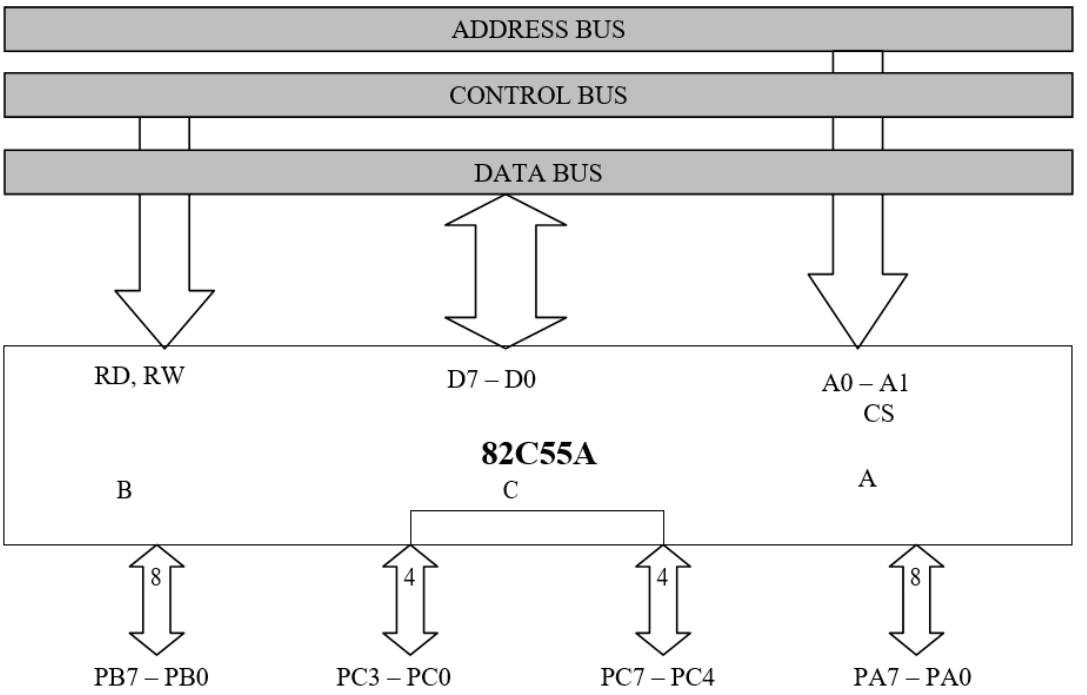
Dalam hal ini, nilai control word yang diperlukan untuk mengatur fungsi port tergantung pada tiga pengaturan yakni group B, group A dan mode set flag. Misal jika diinginkan semua port berfungsi sebagai saluran output maka untuk group B diatur sehingga menghasilkan bit-bit D2D1D0=000, group A menghasilkan D6D5D4D3=0000, dan mode set flag menghasilkan D7=1, sehingga secara keseluruhan pengaturan tersebut menghasilkan control word D7D6D5D4D3D2D1D0=10000000, atau nilai tersebut sama dengan 80 heksadesimal. Dengan demikian untuk mode 0 terdapat 16 kombinasi nilai control word untuk mengatur fungsi port seperti ditunjukkan pada tabel 3.

Tabel 12. Nilai control word yang diperlukan pada pemrograman mode 0 PPI 8255

Fungsi Port				Nilai Control Word dalam Heksadesimal
Port A	**Port C Upper**	**Port B**	**Port C Lower**	
OUTPUT	OUTPUT	OUTPUT	OUTPUT	80
OUTPUT	OUTPUT	OUTPUT	INPUT	81
OUTPUT	OUTPUT	INPUT	OUTPUT	82
OUTPUT	OUTPUT	INPUT	INPUT	83
OUTPUT	INPUT	OUTPUT	OUTPUT	88
OUTPUT	INPUT	OUTPUT	INPUT	89
OUTPUT	INPUT	INPUT	OUTPUT	8A
OUTPUT	INPUT	INPUT	INPUT	8B
INPUT	OUTPUT	OUTPUT	OUTPUT	90
INPUT	OUTPUT	OUTPUT	INPUT	91
INPUT	OUTPUT	INPUT	OUTPUT	92
INPUT	OUTPUT	INPUT	INPUT	93
INPUT	INPUT	OUTPUT	OUTPUT	98
INPUT	INPUT	OUTPUT	INPUT	99
INPUT	INPUT	INPUT	OUTPUT	9A
INPUT	INPUT	INPUT	INPUT	9B

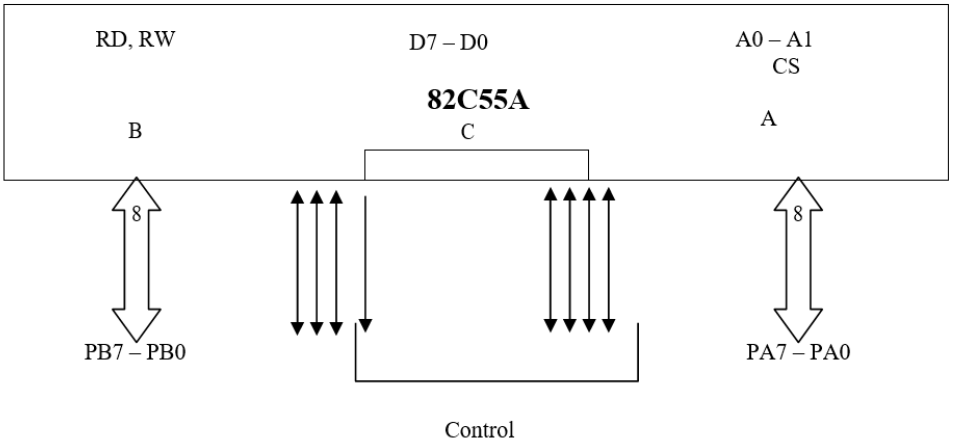
Konfigurasi fungsi ini menghasilkan operasi masukan/keluaran yang sederhana untuk masing-masing port. Apabila port A dan port B dioperasikan dalam mode 0, maka dua bagian dalam port C bisa digunakan sebagai 2 buah port 4 bit. Fungsi dasar mode 0 :

1) Dua port 8 bit dan dua port 4 bit.
2) Semua port dapat dipakai sebagai masukan atau keluaran.
3) Keluaran ditahan (latch), sedangkan masukan tidak.



Gambar 44. Mode operasi 0

b. Mode 2 (strobe bidirectional bus I/O)
Hanya port A yang bisa dioperasikan dalam mode 2. Dalam mode 2 port A digunakan sebagai port transfer data dengan handshake dua arah, artinya data dapat dikeluarkan atau dimasukan pada mode ini untuk memperluas sistem bus ke mikroprosesor tambahan atau untuk mentranfer data dari atau ke kartu kendali sebuah floppy disk. Apabila port A dioperasikan dalam mode 2, maka PC3 sampai PC7 diginakan sebagai jalur handshake untuk port A, sedangkan 3 buah port C lainnya dapat digunakan untuk masukan atau keluaran, apabila port B dalam mode 0. Apabila port B dalam mode 1, maka jalur port C tersebut digunakan sebagai jalur handshake.



Gambar 45. Mode operasi 2

c. Pemrograman PPI 8255

Pemilihan mode-mode operasi dilakukan dengan cara memberikan data kendali ke control register 8255. pemberian data kendali atau proses inisialisasi dilakukan pada saat masukan A0 = A1 tinggi (1). Setelah mengalami proses inisialisasi, maka mode operasi dari setiap port 8255 tidak akan berubah sampai mengalami proses inisialisasi lagi atau direset.

Inti dari pengendalian terdapat pada PPI 8255, yaitu terpadu masukan keluaran yang dapat diprogram. Ketiga buah port 8255 digunakan semuanya.

Penggunaan untuk PPI 8255 di dalam pembahasan ini adalah :

1. Port A digunakan sebagai port keluaran (mode 0)
2. Port B digunakan sebagai port keluaran (mode 0)
3. Port C digunakan sebagai port keluaran (mode 0)

Adapun program yang akan mengekskusinya PPI 8255 yang Port A, B, dan C sebagai Out put adalah sebagai berikut :

```
PPIA    EQU   $6000     Alamat Akses Port A 8255
PPIB    EQU   $6001     Alamat Akses Port B 8255
PPIC    EQU   $6002     Alamat Akses Port C 8255
PPIREG  EQU   $6003     Alamat Akses Register Pengendali

        ORG   $3000     Alamat awal RAM EVBU 68HC11
        LDAA  #$80      Data Masukan 80 Heksa
        STAA  PPIREG
```

Mulai masuk program utama sesuai dengan yang kita inginkan

D. Tugas Pendahuluan

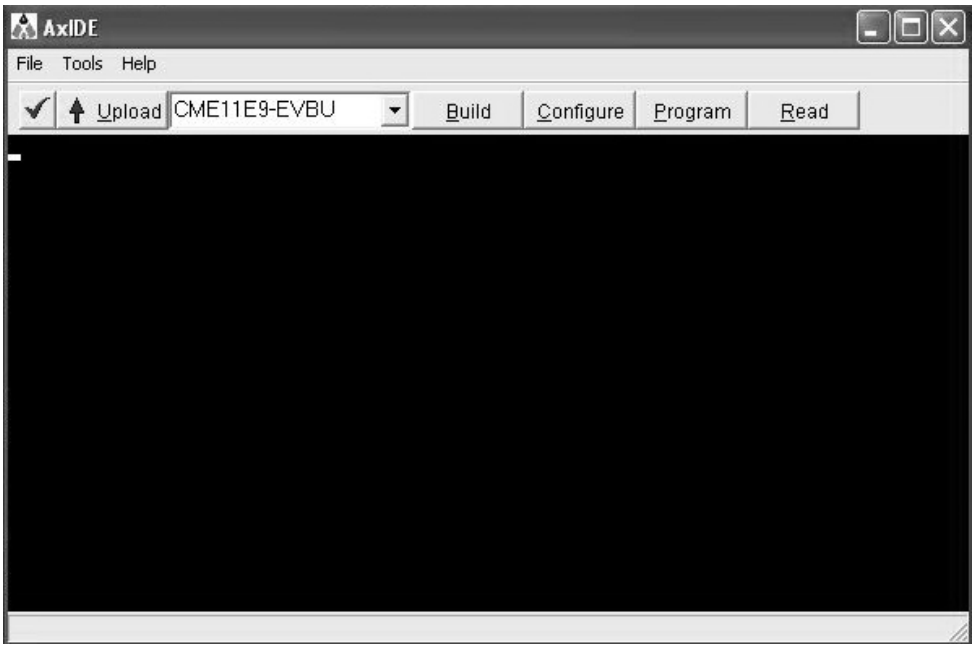
Apakah yang anda ketahui tentang mode operasi sigle chip? Resources M68HC11 apa sajakah yang dapat diakses pada mode ini?

E. Alat-alat dan Langkah Percobaan

1. Satu unit sistem 68HC11 EVBU
2. Satu unit modul Basic I/O
3. Satu unit catu daya
4. Software AxIDE
5. Software Programmer's Notepad

F. Persiapan Percobaan

1. Pasang Kabel Serial Port COM pada Host PC.
2. Atur posisi jumper pada EVBU agar beroperasi pada mode single-chip dan Trace enable, yaitu dengan memasang jumper J3 dan J6.
3. Hubungkan Kabel Serial yang dengan EVBU.
4. Hubungkan Catu Daya dengan EVBU.
5. Aktifkan/Run program AxIDE.



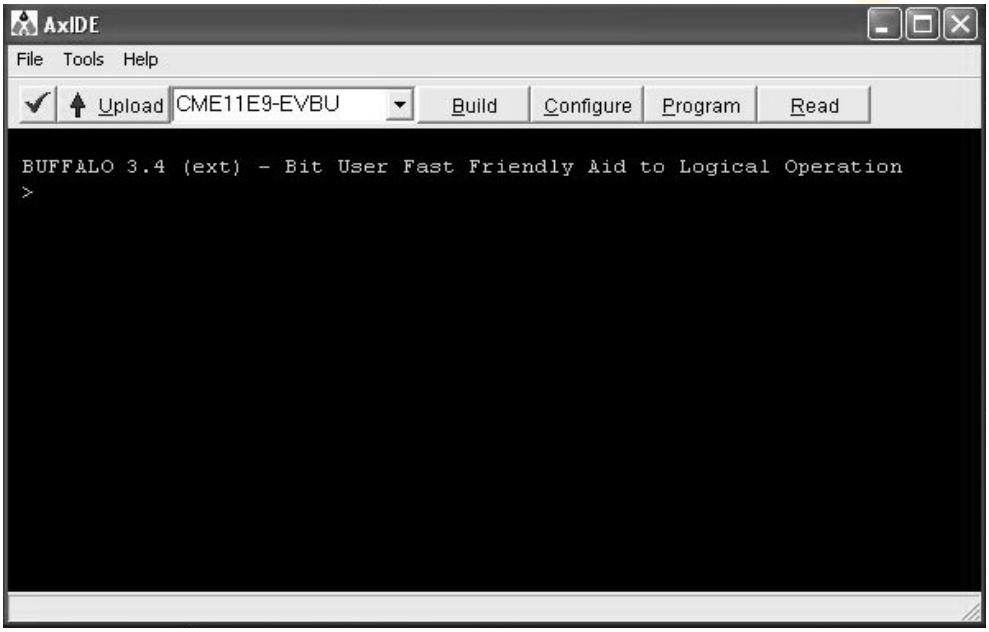
Gambar 46. Tampilan Program AxIDE

6. Click icon Check (Centang) atau melalui menu **File>Option**, pastikan setting komunikasi yang digunakan adalah sebagai berikut :

Tabel 13. Setting Komunikasi Program AxIDE

Port	COM1/COM2	Handshaking	Information
Baud Rate	9600		
Parity	None	Xon/Xoff	OFF
Data Bits	8	Rts/Cts	OFF
Stop Bits	1	Dtr/Dsr	OFF

   Jika belum sesuai lakukan pengaturan agar diperoleh setting seperti tampak pada tabel di atas.
7. Tekan tombol Reset pada EVBU.
8. Pastikan muncul tampilan BUFFALO pada terminal window AxIDE, yang menunjukkan bahwa EVBU dan komputer telah terhubung melalui Port Serial.
9. Tekan Enter agar anda berada pada prompt BUFFALO sebagai berikut:



Gambar 47. Tampilan Prompt BUFFALO Program AxIDE

Keterangan lebih lengkap ada pada petunjuk penggunaan software halaman 01 s/d 08.

10. Mengirimkan Data ke Port B 8255

Hubungkan kabel IDC 26pin dari EVBU ke modul BASIC I/O. Dengan langkah ini berarti anda telah memasang 8 buah LED pada pin port B 8255, dan 8 buah saklar pada pin port C 8255.

Menggunakan software Programmer's Notepad, tulis program berikut ini:

```
PPIB    EQU   $6001    Alamat Akses Port B 8255
PPIC    EQU   $6002    Alamat Akses Port C 8255
PPIREG  EQU   $6003    Alamat Akses Register Pengendali

        ORG   $3000    Alamat awal RAM EVBU 68HC11
        LDAA  #$80     Data Masukan 80 Heksa
        STAA  PPIREG
        LDAA  #$FF
        STAA  PPIB
        JMP   $3000
```

Simpan dengan nama SBM1B1.asm. Lakukan assembly melalui program AxIDE, Download hasil assembly ke EVBU. Jalankan program dengan perintah GO 3000. Amati LED yang terhubung ke port B! Apakah keadaan nyala LED sesuai dengan data %1111

1111? Bagaiamanakah caranya agar LED bias menyala bergantian(seperti berjalan) dari Kanan ke Kiri atau sebaliknya? Buatlah programnya!

11. Membaca Data di Port C 8255 dan Mengirim ke Port B 8255.
    Hubungkan kabel IDC 26pin dari EVBU ke modul BASIC I/O. Dengan langkah ini berarti anda telah memasang 8 buah LED pada pin port B 8255, dan 8 buah saklar pada pin port C 8255. Menggunakan software Programmer's Notepad, tulis program berikut ini:

```
PPIB    EQU   $6001    Alamat Akses Port B 8255
PPIC    EQU   $6002    Alamat Akses Port C 8255
PPIREG  EQU   $6003    Alamat Akses Register Pengendali

        ORG   $3000    Alamat awal RAM EVBU 68HC11
        LDAA  #$89     Data Masukan 89 Heksa
        STAA  PPIREG
        LDAA  PPIC
        STAA  PPIB
        JMP   $3000
```

    Simpan dengan nama SBM1B2.asm. Lakukan assembly melalui program AxIDE, Download hasil assembly ke EVBU. Jalankan program dengan perintah GO 3000. Amati yang terjadi pada LED pada I/O Basic, untuk setiap kombinasi DIP Switch!

G. TUGAS AKHIR

Buatlah Program agar Kombinasi tertentu pada DIP Switch mengakibatkan LED menyala semua, atau Menyala dari Kanan ke Kiri, atau Menyala dari Kiri ke Kanan, atau Menyala semua dan berkedip.