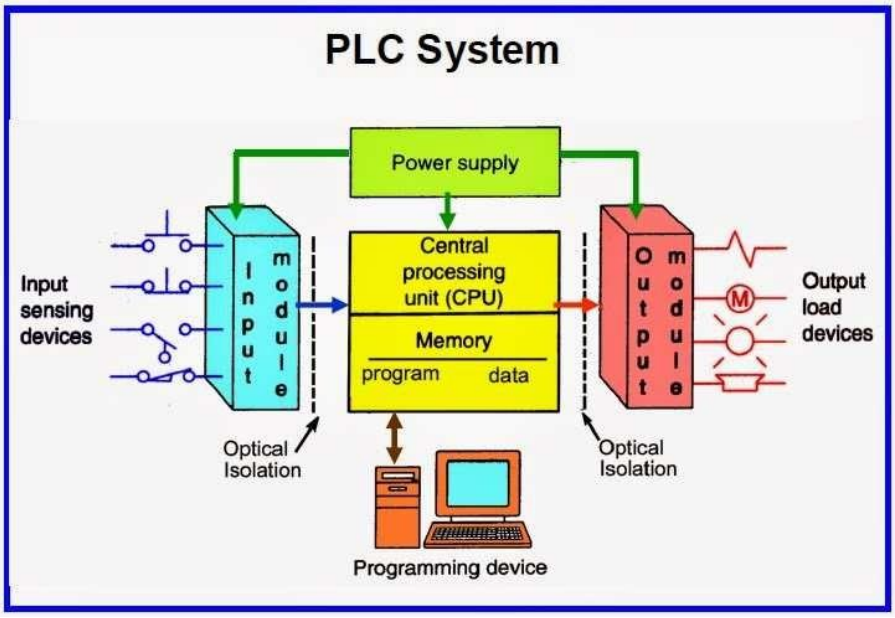
MODUL PRAKTIKUM

OTOMASI - II
(PROGRAMMABLE LOGIC CONTROLLER)
= PLC =



Disusun Oleh :
TAUFIK HIDAYAT, ST,MT

INSTITUT SAINS DAN TEKNOLOGI NASIONAL
JURUSAN TEKNIK ELEKTRO
2022

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Kasih, karena dengan penyertaan dan tuntunan-Nya maka penulis dapat menyelesaikan modul ini. Mata Kuliah praktek adalah mata kulliah praktikum dari lanjutan dari Mata Kuliah Otomasi I yang dimasukkan dalam Kurikulum Program Studi S1 Teknik Elektro dengan isi materi Elektropneumatik dan PLC (*Programmable Logic Controller*).

Materi dalam Mata Kuliah PLC ini tentang rangkaian kelistrikan untuk menjalankan rangkaian sistem Pneumatik maupun rangkaian sistem hidrolik dengan bantuan *software* "***FluidSIM***" dari FESTO, dan PLC OMRON dengan bantuan *software* "***CX-Programmer***".

Modul Praktikum ini dibuat untuk mahasiswa khususnya Jurusan Elektro, agar kiranya mendapatkan tambahan ilmu untuk merancang dan membuat suatu rangkaian sistem Otomasi Industri yang terpadu dalam mendukung Era Generasi 4.0

Jakarta, Nopember 2022 Penulis,

Taufik Hidayat, ST., MT

DAFTAR ISI

COVER i
HALAMAN PENGESAHAN ii
KATA PENGANTAR iii
DAFTAR ISI iv
PENGENALAN HARDWARE 1
MEMORY CJ1M CPU 11 2
INSTRUKSI-INSTRUKSI DASAR PADA PLC CJ1M CPU11 4
- Instruksi Input 4
- Instruksi *Output* 5
- Instruksi *Timer* dan *Counter* 8
- Instruksi Perbandingan Data 9
- Instruksi Pemindah Data 11
- Instruksi Penggeseran Data 12
- Instruksi Increment dan Decrement 13
- Instruksi Aritmetika 14

PEMOGRAMAN PLC CJ1M CPU11 menggunakan LSS (*Ladder Support Software*) *CX-Programmer* 16
- Open *CX-Programmer* 16
- Pemilihan Jenis PLC dan Koneksi PLC 17
- Contoh Pembuatan Program 19
- Men-*transfer* dan Menjalankan Program 20
- Contoh Program 24
- Soal-soal Latihan 26

MODUL PRAKTIKUM PLC
(*PROGRAMMABLE LOGIC CONTROLLER*)

I. PENGENALAN *HARDWARE*

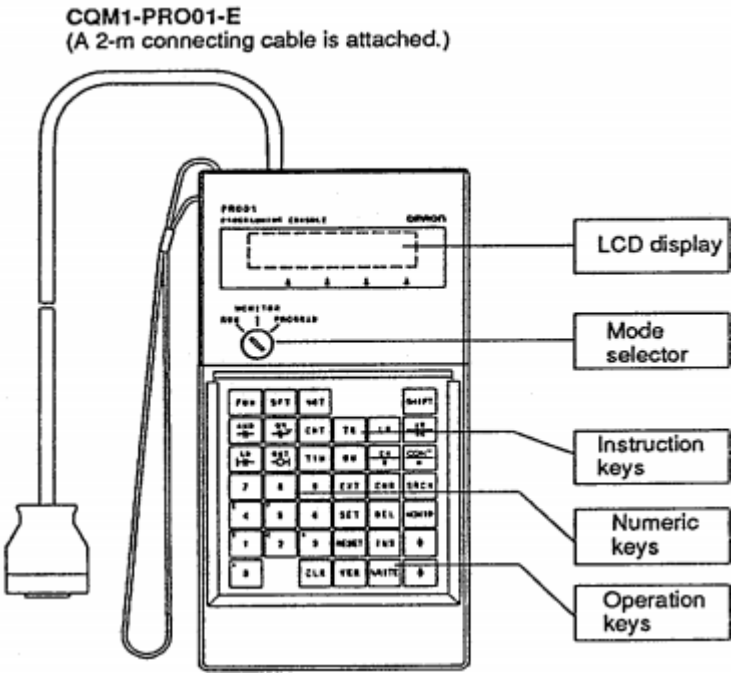
PLC yang digunakan untuk praktikum ini adalah PLC Omron seri CJ1M CPU11, PLC ini mempunyai alamat eksternal *Input* dari 0.00 sampai 0.15, dan alamat eksternal *Output* dari 1.00 sampai 1.15. Disebut alamat eksternal karena *input* dan *output* ini terhubung dengan "dunia luar" PLC. Eksternal *input* terhubung dengan sensor, atau peralatan-peralatan masukan yang lain. Eksternal *output* terhubung dengan aktuator.

Input 0.00 sampai 0.15 dihubungkan dengan *Pushbutton*, *input* 0.14 dan 0.15 juga terhubung dengan *toggle switch*, sedangkan *output* 1.00 sampai 1.15 dihubungkan dengan lampu. *Output* 1.14 dan 1.15 juga dihubungkan dengan *buzzer*.

Alamat *internal* yang biasa digunakan adalah dari *channel* 200 sampai *channel* 220 (dalam bentuk *bit* adalah dari alamat 200.00 sampai 220.15), yang semuanya bisa diakses dalam tiap *bit*-nya. Satu *channel* atau *word* memiliki lebar data sebanyak 16 *bit*.

Timer dan *Counter* memiliki daerah memori yang sama yaitu memori T/C yang mempunyai alamat dari 0000 sampai 4095.

Pembuatan program menggunakan LSS (*Ladder Support Software*) *CX-Programmer* yang dihubungkan secara langsung dengan PLC menggunakan koneksi serial RS232. Selain menggunakan *CX-Programmer* PLC CJ1M CPU11 dapat juga diprogram menggunakan *Programming Console*.



II. MEMORI CJ1M CPU11.

Memori pada CJ1M CPU11 dibagi menjadi beberapa bagian, yang masing-masing bagian mempunyai fungsi tertentu pada pemrograman. Memori tersebut dipetakan menurut fungsinya di CJ1M CPU11. Berikut akan dibahas peta memori yang sering digunakan pada pemrograman.

2.1 Memori CIO (CIO0000 sampai CIO6143)

Memori CIO (*Core I/O Area*) ini banyak digunakan pada program. Khusus untuk CIO0000 digunakan untuk alamat *input* eksternal dan CIO0001 digunakan untuk alamat *output* eksternal. Sedangkan yang lain digunakan sebagai memori bantu pada pembuatan program, yang akan menyimpan sementara suatu kondisi. Jika *power* pada PLC dimatikan, semua kondisi pada memori CIO akan kembali menjadi nol.
Memori CIO bisa diakses dalam bentuk *word* atau dalam bentuk *bit* (0000.00 sampai 6143.15).

2.2 Memori A (A000 sampai A959)

Memori A (*Auxiliary area*) jarang digunakan secara langsung, baik dalam bentuk *word* atau dalam bentuk *bit*. Hal ini karena memori A lebih banyak menyimpan kondisi jika suatu logika terpenuhi. Memori ini tidak dapat digunakan untuk menyimpan data.
Memori A bisa diakses dalam bentuk *word* atau dalam bentuk *bit* (A000.00 sampai A959.15).

2.3 Memori H (H000 sampai H511)

Memori H (*Holding area*) digunakan untuk menyimpan data. Data akan disimpan walaupun *power* PLC dimatikan.
Memori H bisa diakses dalam bentuk *word* atau dalam bentuk *bit* (H000.00 sampai H511.15).

2.4 Memori T (T0000 sampai T4095)

Memori T (*Timer area*) digunakan sebagai alamat *timer*. Memori ini tidak bisa digunakan untuk menyimpan data, tapi bisa digunakan data yang ada pada *timer* untuk disimpan di memori lain.
Memori T hanya bisa diakses dalam bentuk *word*.

2.5 Memori C (C0000 sampai C4095)

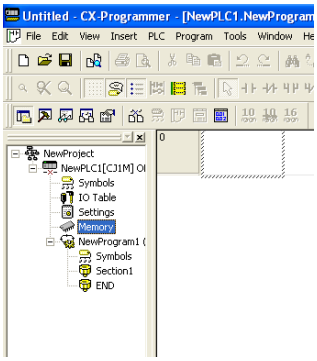
Memori C (*Counter area*) digunakan sebagai alamat *Counter*. Seperti memori T, memori ini tidak bisa digunakan untuk menyimpan data, tetapi bisa diambil data yang ada pada *Counter* untuk disimpan di memori lain.

2.6 Memori D (D00000 sampai D32767)

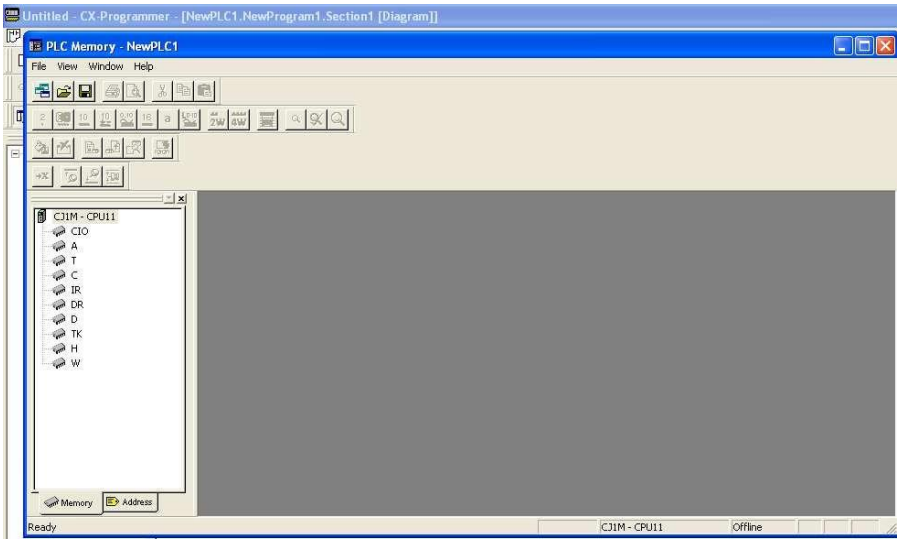
Merupakan memori yang terbanyak yang ada di PLC CJ1M CPU11. Memori ini digunakan untuk menyimpan data, sering juga disebut memori DM (*Data Memory*).

Memori ini tidak bisa diakses langsung dalam bentuk *bit*, tetapi tiap *bit* dari memori ini bisa gunakan untuk menyimpan data. Sama seperti memori H, data yang tersimpan akan terus disimpan walaupun *power* PLC dimatikan. Tidak semua bagian memori ini dapat digunakan, karena ada beberapa bagian memori ini yang digunakan untuk menyimpan pengaturan PLC. Memori-memori yang lain dapat dilihat pada tampilan dengan cara berikut berikut:

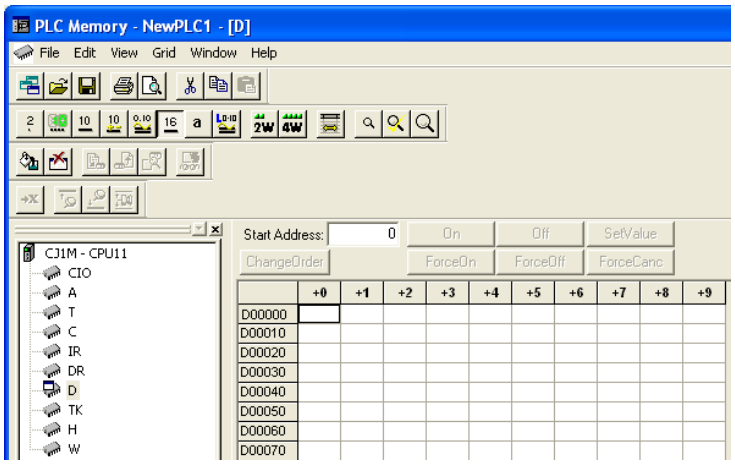
Klik ganda pada Memory



Akan muncul menu PLC Memory seperti berikut:



Kemudian pilih memori yang akan dilihat. Pada contoh berikut akan ditunjukkan memori D yang akan dilihat isi memorinya. Klik ganda pada memori D seperti pada contoh berikut:



Untuk memori-memori yang lain, cara membukanya sama dengan contoh yang ada.

III. INSTRUKSI-INSTRUKSI DASAR pada PLC CJ1M CPU11.

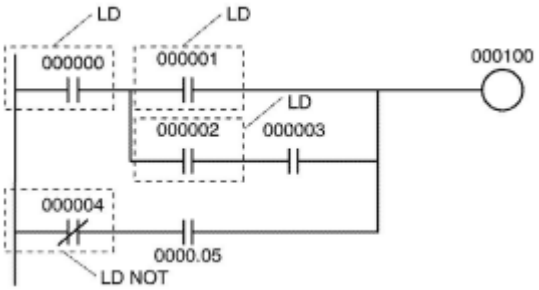
1.1 Instruksi *Input*

Merupakan instruksi masukan yang membentuk logika, kondisi *on/off* input menunjukkan kondisi *on/off bit* yang merupakan alamat *input* tersebut. Beberapa jenis instruksi *input* adalah:

Load: LD

Input dilambangkan dengan perintah LD, akan *on* atau *off* sesuai dengan kondisi *input* tersebut. Simbol instruksi LD adalah:

Contoh penggunaan LD pada suatu program:



Load Not: LD NOT

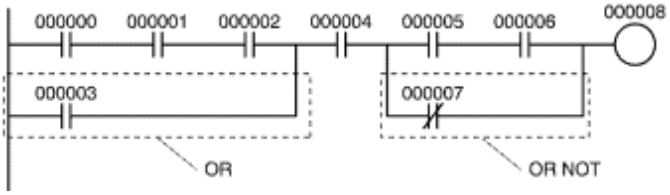
Input dilambangkan dengan perintah LD NOT, akan *on* atau *off* sesuai dengan kondisi *input* tersebut. Perintah LD NOT mempunyai kondisi yang berkebalikan dengan perintah LD. Simbol instruksi LD NOT adalah:

AND dan **AND NOT: AND** dan **AND NOT**

Menggabungkan dua atau lebih *input* secara seri. Dapat divariasikan dengan perintah AND NOT.

OR dan **OR NOT: OR** dan **OR NOT**

Menggabungkan dua atau lebih *input* secara paralel. Dapat divariasikan dengan perintah OR NOT. Contoh penggunaan dalam suatu program:

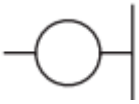


1.2 Instruksi *Output*

Adalah instruksi yang merupakan hasil dari proses logika dari *input-input* yang ada, dan akan menjadikan *bit-bit* tertentu yang merupakan alamat *output* akan *on* atau *off*.

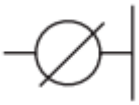
OUTPUT: OUT

Instruksi ini akan mejadikan sebuah *bit* menjadi *on* selama *input* diaktifkan. Simbol instruksi OUT sebagai berikut:

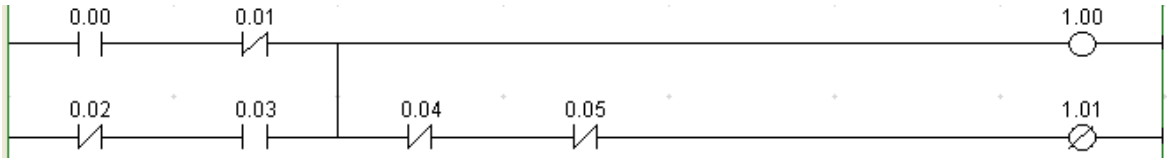


OUTPUT NOT: OUT NOT

Instruksi ini akan menjadikan sebuah *bit* menjadi *off* selama *input* diaktifkan. Simbol instruksi OUT NOT sebagai berikut:

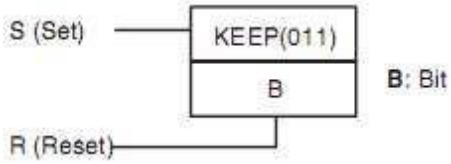


Contoh aplikasi program untuk instruksi LD, LD NOT, AND, OR, AND NOR, OR NOT, OUT, dan OUT NOT pada PLC CJ1M CPU11.



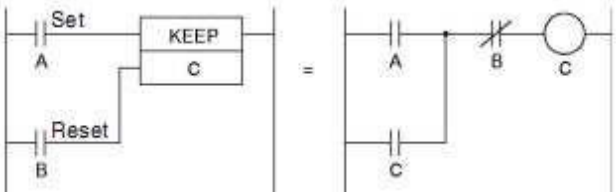
KEEP: KEEP(011)

Instruksi ini akan menahan sebuah *bit* menjadi *on* sampai *input reset* diaktifkan. Simbol instruksi KEEP(011) sebagai berikut:



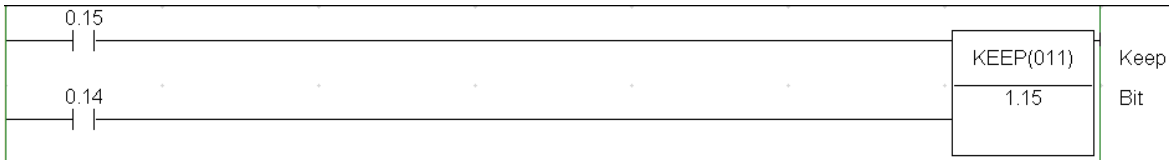
Instruksi KEEP(011) mempunyai fungsi yang hampir sama dengan logika *self holding.*

Kondisi *bit* pada instruksi KEEP(011) yaitu C akan *off* jika *input reset* diaktifkan.



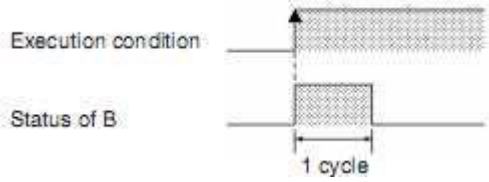
Self Holding (Pengunci)

Contoh instruksi KEEP(011) pada PLC CJ1M CPU11:

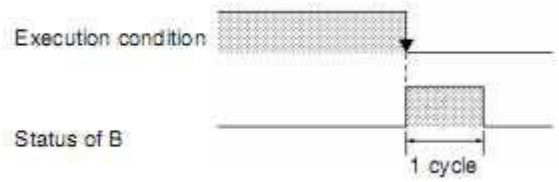


DIFFERENTIATE UP/DOWN: DIFU(013) dan **DIFD(014)**

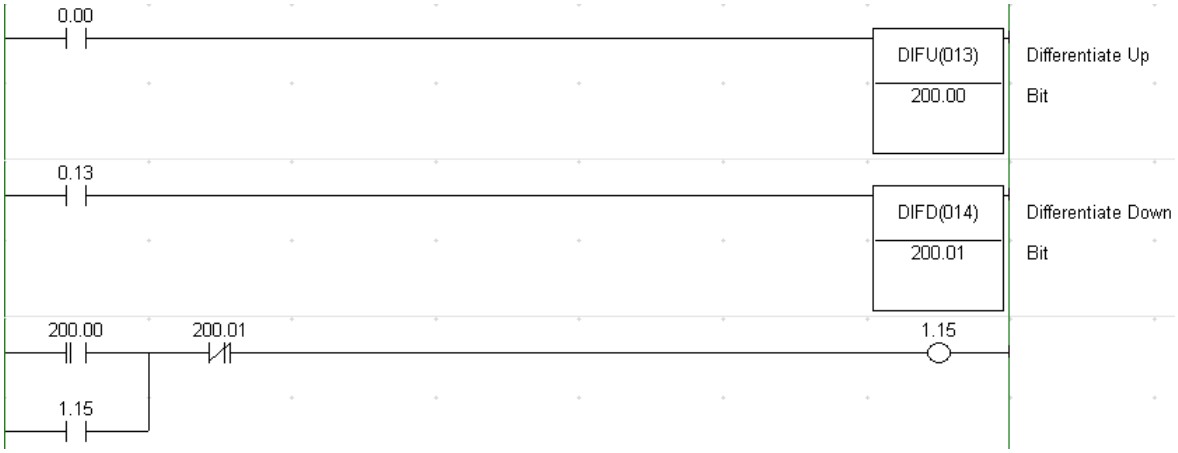
DIFU(013) akan membuat sebuah *bit* menjadi *on* selama satu siklus ketika kondisi *input* dari *off* menuju *on* (*rising edge*).



DIFD(014) akan membuat sebuah *bit* menjadi *on* selama satu siklus ketika kondisi *input* dari *on* menuju *off* (*falling edge*).

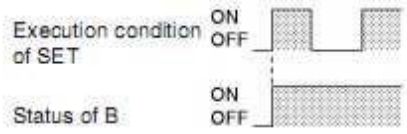


Contoh instruksi DIFU(013) dan DIFD(014) pada PLC CJ1M CPU11:

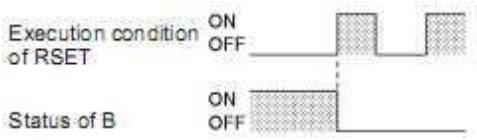


SET dan **RESET: SET** dan **RSET**

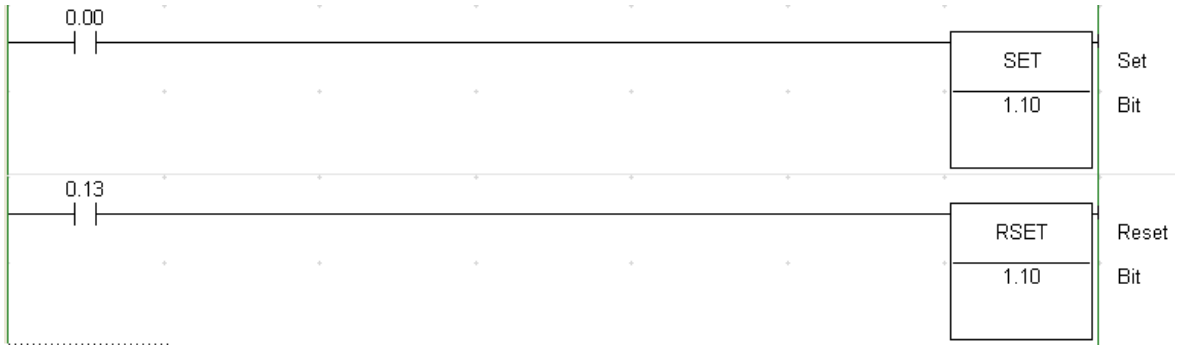
SET akan membuat sebuah *bit* menjadi *on* selamanya jika kondisi eksekusinya *on*, dan tetap *on* walaupun *input*nya *off*.



RSET akan membuat sebuah *bit* menjadi *off* jika kondisi eksekusinya *on*.



Contoh instruksi SET dan RSET pada PLC CJ1M CPU11:



1.3 Instruksi *Timer* dan *Counter*

Adalah instruksi yang digunakan untuk meng-*handle* pewaktu dan pencacah peristiwa. Pada PLC CJ1M CPU11 ada banyak jenis *Timer* dan *Counter*, tetapi tidak dibahas semua.

TIMER: TIM(XXXX)

Adalah instruksi yang digunakan untuk meng-*handle* fungsi pewaktu. Bentuk fungsi *timer* adalah sebagai berikut:

PV refresh method	Symbol	Operands
BCD	TIM N S **N**: Timer number **S**: Set value	N: 0000 to 4095 (decimal) S: #0000 to #9999 (BCD)

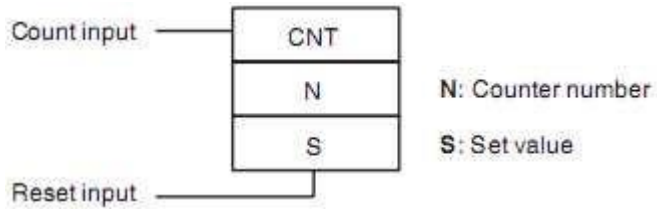
Ketika *timer* mendapat *input*, maka *set value timer* akan berkurang, sampai menjadi nol. Jika sudah nol maka *timer* tersebut akan aktif.

Nilai *set value timer* diisi dengan bilangan BCD, dimana nilai pewaktunya adalah bilangan BCD tersebut dikali 0.1 detik.

Contoh instruksi *timer* pada PLC CJ1M CPU11:

COUNTER: CNT(XXXX)

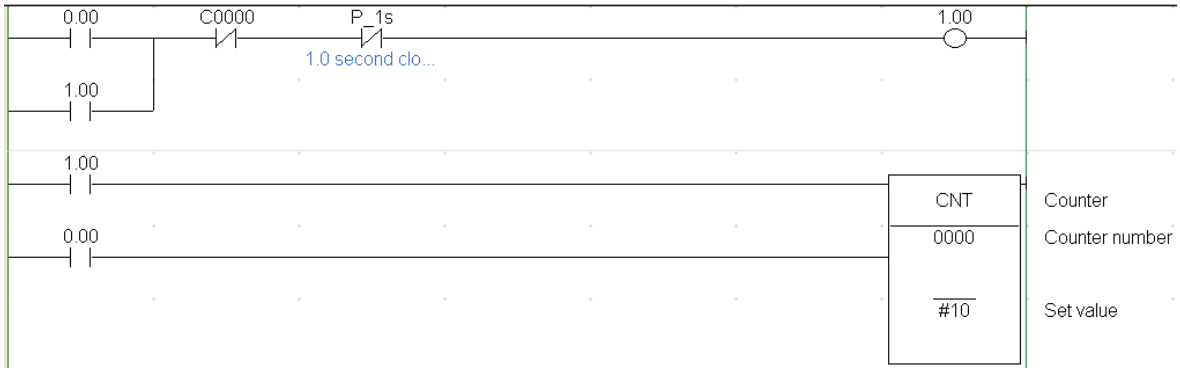
Adalah instruksi yang digunakan untuk meng*handle* pencacah peristiwa. Bentuk fungsi *Counter* adalah sebagai berikut:



Ketika *Counter* mendapat *input*, maka *set value counter* akan berkurang, sampai menjadi nol. Jika sudah nol maka *Counter* tersebut akan aktif.

Nilai *set value counter* diisi dengan bilangan BCD, dimana nilai pencacahnya adalah sebanyak bilangan BCD yang dimasukkan. (BCD : Binary Code Decimal).

Contoh instruksi *Counter* pada PLC CJ1M CPU11:



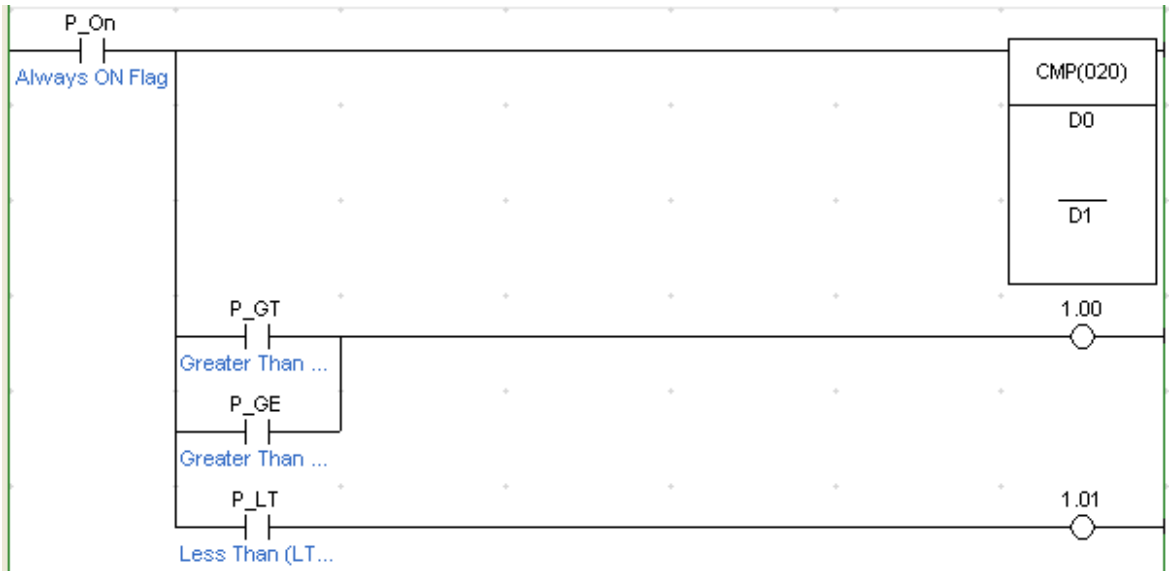
1.4 Instruksi Perbandingan Data

Adalah instruksi yang akan membandingkan dua data, baik data yang tersimpan dalam suatu memori atau data berupa konstanta dalam bentuk BCD, dan hasil perbandingan tersebut akan mengaktifkan sebuah *bit* pada *Auxiliary Area*.

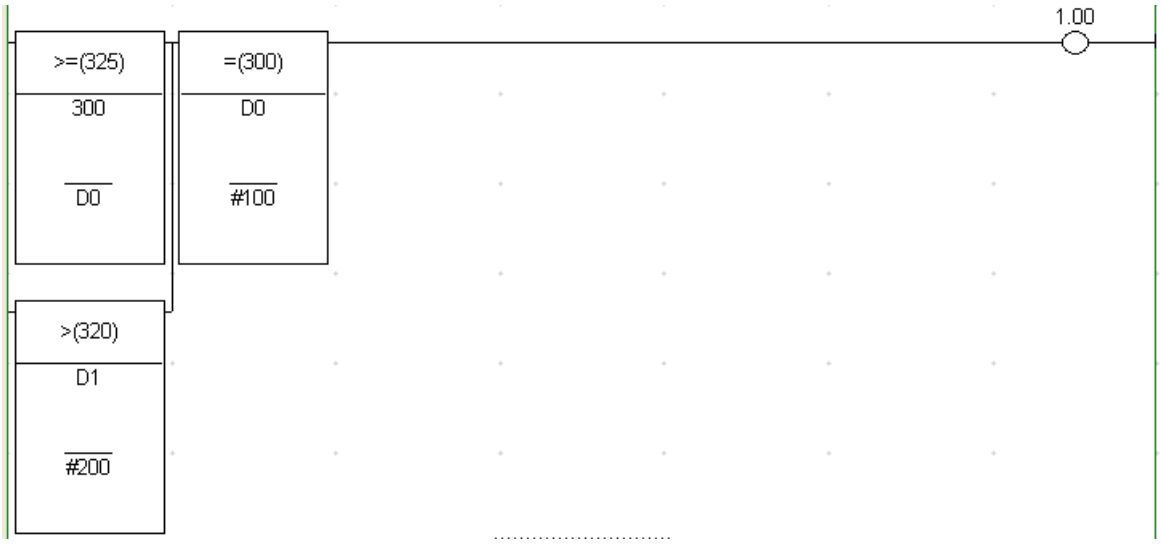
COMPARE: CMP(020)

Bentuk instruksi CMP(020) sebagai berikut:

Contoh penggunaan instruksi CMP(20) pada suatu program:



Selain digunakan sebagai *output*, instruksi perbandingan dapat juga digunakan sebagai *input*. Berikut beberapa bentuk perbandingan sebagai *input*. Contoh instruksi perbandingan yang digunakan sebagai *input* pada suatu program:



Perbandingan-perbandingan yang sebagai *input* mungkin dilakukan ditabelkan seperti berikut:

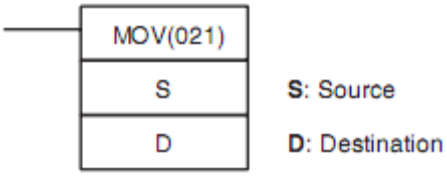
Symbol		Option (data format)	Option (data length)
=	(Equal)	None: Unsigned data	None: One-word data
<>	(Not equal)	S: Signed data	L: Double-length data
<	(Less than)		
<=	(Less than or equal)		
>	(Greater than)		
>=	(Greater than or equal)		

1.5 Instruksi Pemindah Data

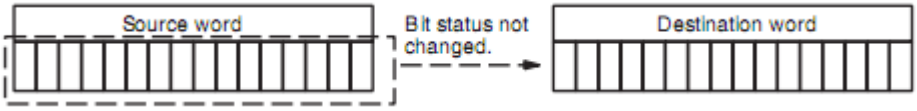
Adalah instruksi untuk menyalin data berupa isi suatu memori atau konstanta, ke suatu alamat memori yang ditentukan.

MOVE: MOV(021)

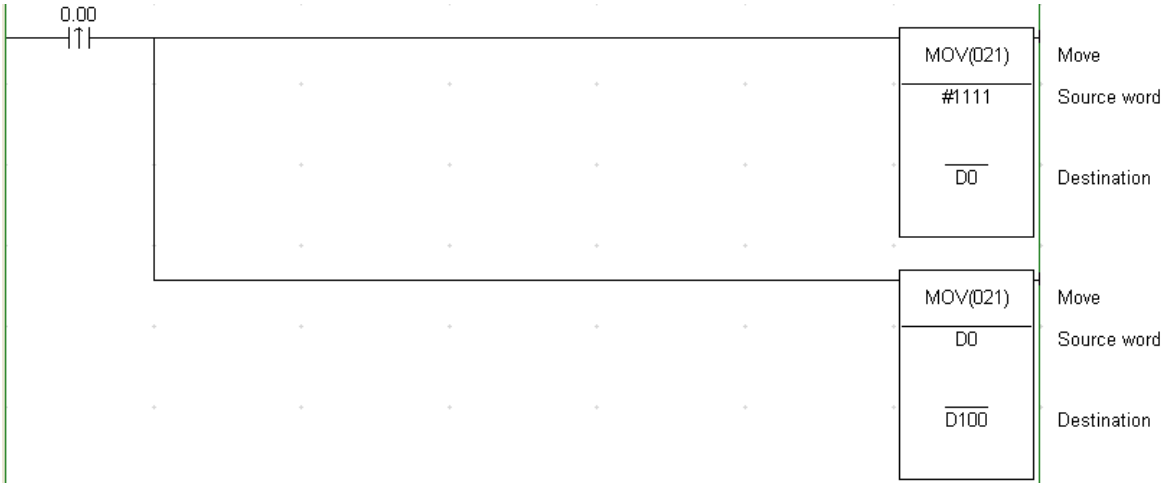
Instruksi MOV(021) akan memindahkan suatu data dari suatu tempat ke memori yang ditentukan. Bentuk instruksi MOV(021) adalah sebagai berikut:



Jika instruksi MOV(021) mendapatkan *input*, maka semua data dari *source* akan disalin seluruhnya pada *destination*. Setelah penyalinan tersebut, data yang ada pada *source* tidak hilang.



Contoh instruksi MOV(020) pada PLC CJ1M CPU11:

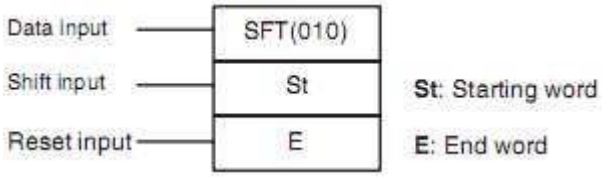


3.5 Instruksi Penggeseran Data

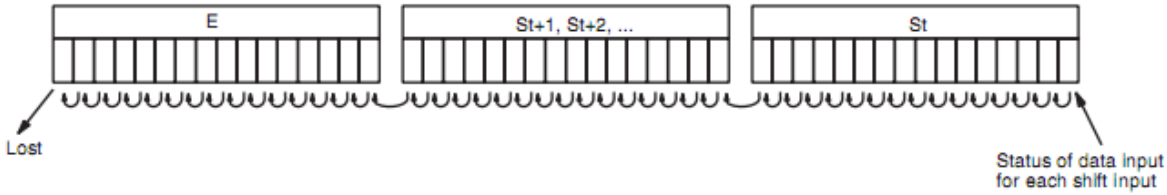
Instruksi penggerseran data akan menggeser data dari *bit* terkecil menuju ke *bit* yang lebih besar.

SHIFT REGISTER: SFT(010)

Instruksi SFT(010) akan menggeser data dari suatu memori ke memori berikutnya. Bentuk instruksi SFT(010) sebagai berikut:

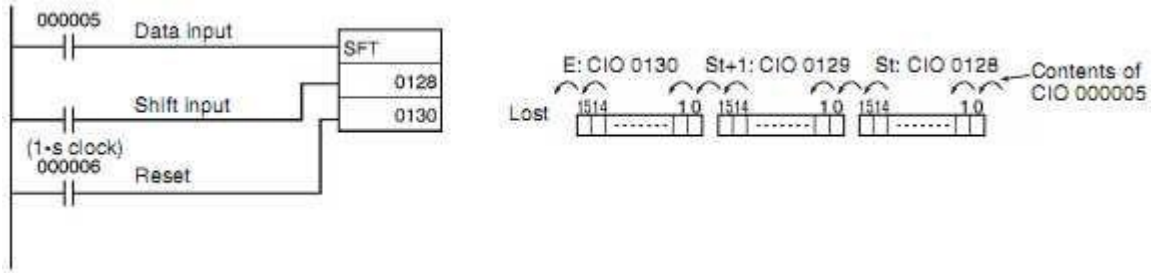


Jika dilihat di dalam memori, bentuk penggeseran yang terjadi dapat digambarkan seperti berikut:

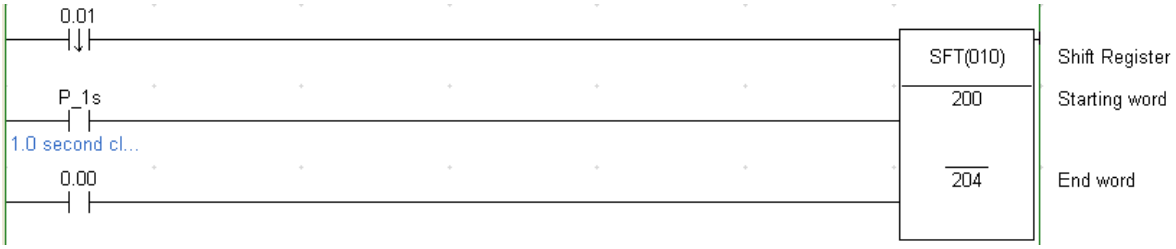


Kecepatan penggeseran data tergantung dari *clock* yang diberikan pada alamat *shift input*. Semakin cepat *clock* pada *shift input*, maka penggeseran data akan semakin cepat. Demikian juga berlaku sebaliknya.

Contoh penggunaan instruksi SFT(010) dalam suatu program:



Contoh instruksi SFT(010) pada PLC CJ1M CPU11:

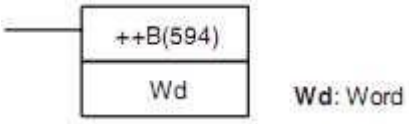


3.7 Instruksi Increment dan Decrement

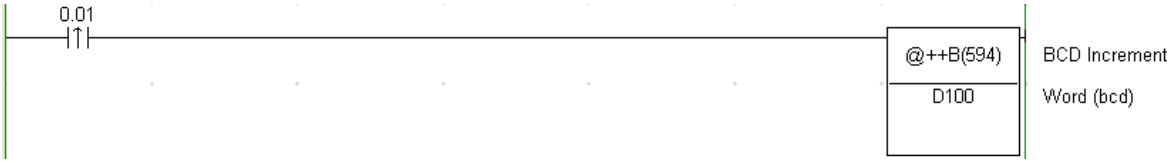
Instruksi *Increment* dan *Decrement* akan menambah dan mengurang data suatu alamat memori dengan 1.

INCREMENT BCD: ++B(594)

Instruksi *Increment* akan menambah data suatu alamat memori dengan 1 pada 4 digit bilangan BCD. Bentuk instruksi *Increment* adalah sebagai berikut:

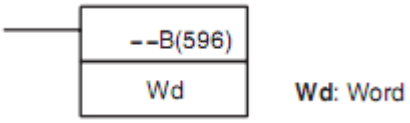


Contoh instruksi ++B(594) pada PLC CJ1M CPU11:

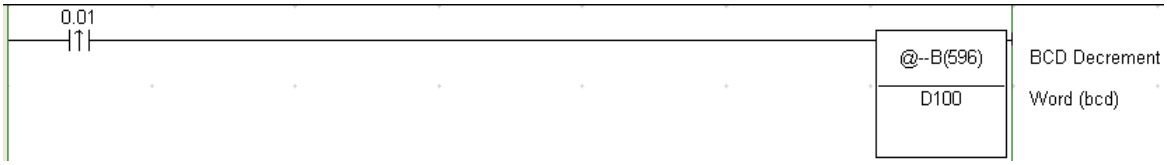


DECREMENT BCD: --B(596)

Instruksi *Decrement* akan mengurang data suatu alamat memori dengan 1 pada 4 digit bilangan BCD. Bentuk instruksi *Decrement* adalah sebagai berikut:



Contoh instruksi –B(596) pada PLC CJ1M CPU11:

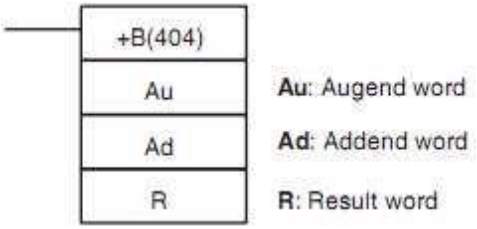


3.8 Instruksi Aritmetika

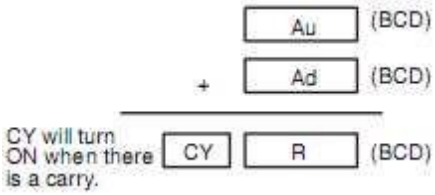
Instruksi-instruksi aritmetika sederhana akan melakukan proses aljabar sederhana jika istruksi tersebut diaktifkan. Instruksi aritmetika yang akan dibahas adalah:

BCD ADD WITHOUT CARRY: +B(404)

Instruksi ini akan melakukan proses penjumlahan data dalam bentuk BCD. Data yang ada atau konstanta yang dibuat di Au (*Augend*) dan Ad (*Addend*) dan diletakkan hasil penjumlahannya pada R (*Result*). Bentuk instruksi ini adalah sebagai berikut:



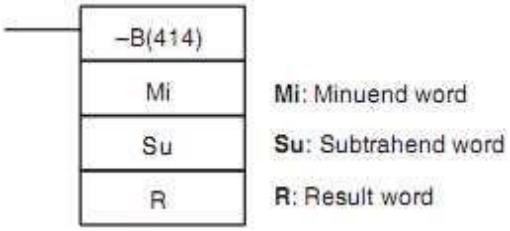
Bentuk operasi yang terjadi pada operasi penjumlahan adalah sebagai berikut:



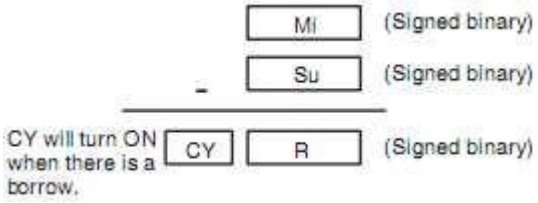
Contoh instruksi +B(404) pada PLC CJ1M CPU11:

BCD SUBTRACT WITHOUT CARRY: -(414)

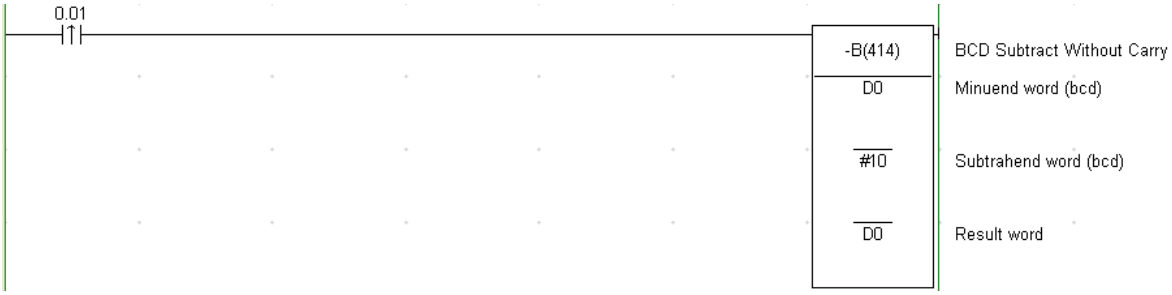
Instruksi ini akan melakukan proses pengurangan data dalam bentuk BCD. Data yang ada atau konstanta yang dibuat di Mi (*Minuend*) dikurangi dengan yang ada di Su (*Subtrahend*) dan hasilnya diletakkan pada R (*Result*). Bentuk instruksi ini adalah sebagai berikut:



Bentuk operasi yang terjadi pada operasi pengurangan adalah sebagai berikut:

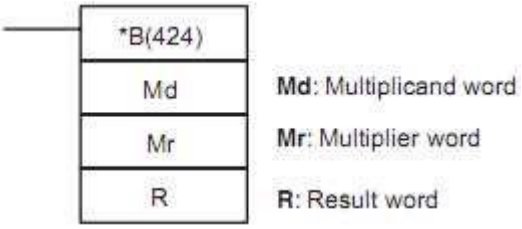


Contoh instruksi –B(414) pada PLC CJ1M CPU11:



BCD MULTIPLY: *B(424)

Instruksi ini akan melakukan proses perkalian data dalam bentuk BCD. Data yang ada atau konstanta yang dibuat di Md (*Multiplicand*) dikalikan dengan data yang ada di Mr (*Multiplier*) dan hasilnya diletakkan pada R (*Result*). Bentuk instruksi ini adalah sebagai berikut:



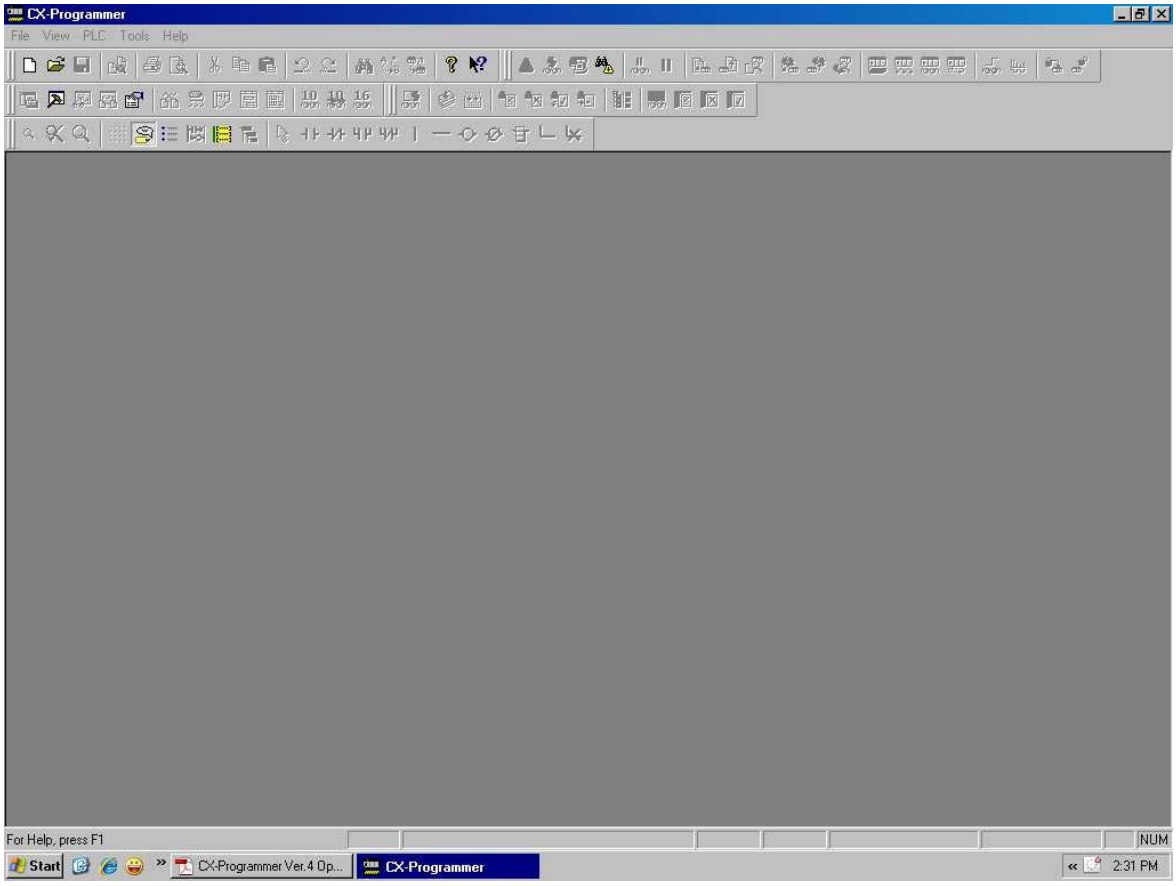
Bentuk operasi yang terjadi pada operasi perkalian adalah sebagai berikut:

Contoh instruksi *B(424) pada PLC CJ1M CPU11:

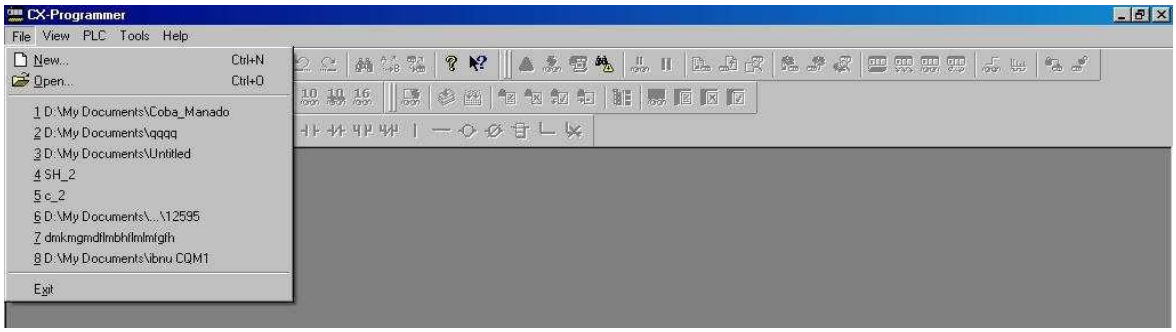
IV. PEMOGRAMAN PLC CJ1M CPU11 menggunakan LSS (*Ladder Support Software*) *CX-Programmer*

4.1 Open *CX-Programmer.*

CX-Programmer dapat dibuka di Main Windows tekan **Start - All Program – OMRON - CX-Programmer - CX-Programmer**. Saat pertama kali membuka *CX-Programmer*, akan terbuka jendela seperti berikut:

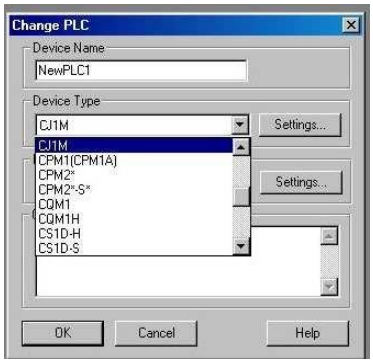


Dengan menggunakan *pointer mouse*, tekan **File-New** untuk mulai membuat program.



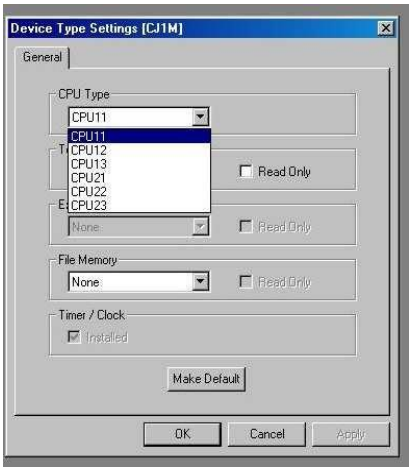
4.2 Pemilihan Jenis PLC dan Koneksi PLC

Pemilihan jenis PLC dan koneksi PLC dibuat setelah proses pembuatan program baru. Akan muncul jendela berikut:

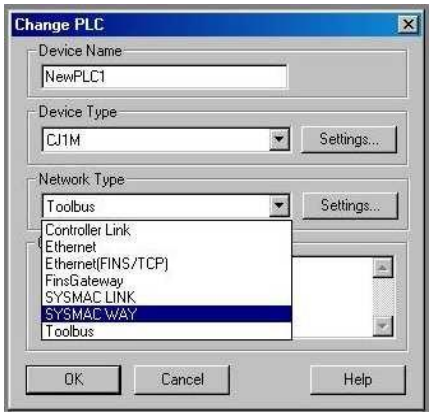


Pada **Change PLC**, **Device Name** nama *default* adalah **NewPLC1** dapat diubah dengan nama apapun sesuai dengan keinginan pembuat program.

Setelah itu dilanjutkan dengan pemilihan PLC yang sesuai dengan yang digunakan. Dalam hal ini digunakan seri CJ1M. Pada kotak ***Setting***, pilih CPU PLC yang sesuai, dapat dilihat pada bodi PLC. PLC yang digunakan di sini menggunakan CPU11. Setelah proses pemilihan dilakukan, tekan OK seperti gambar berikut.

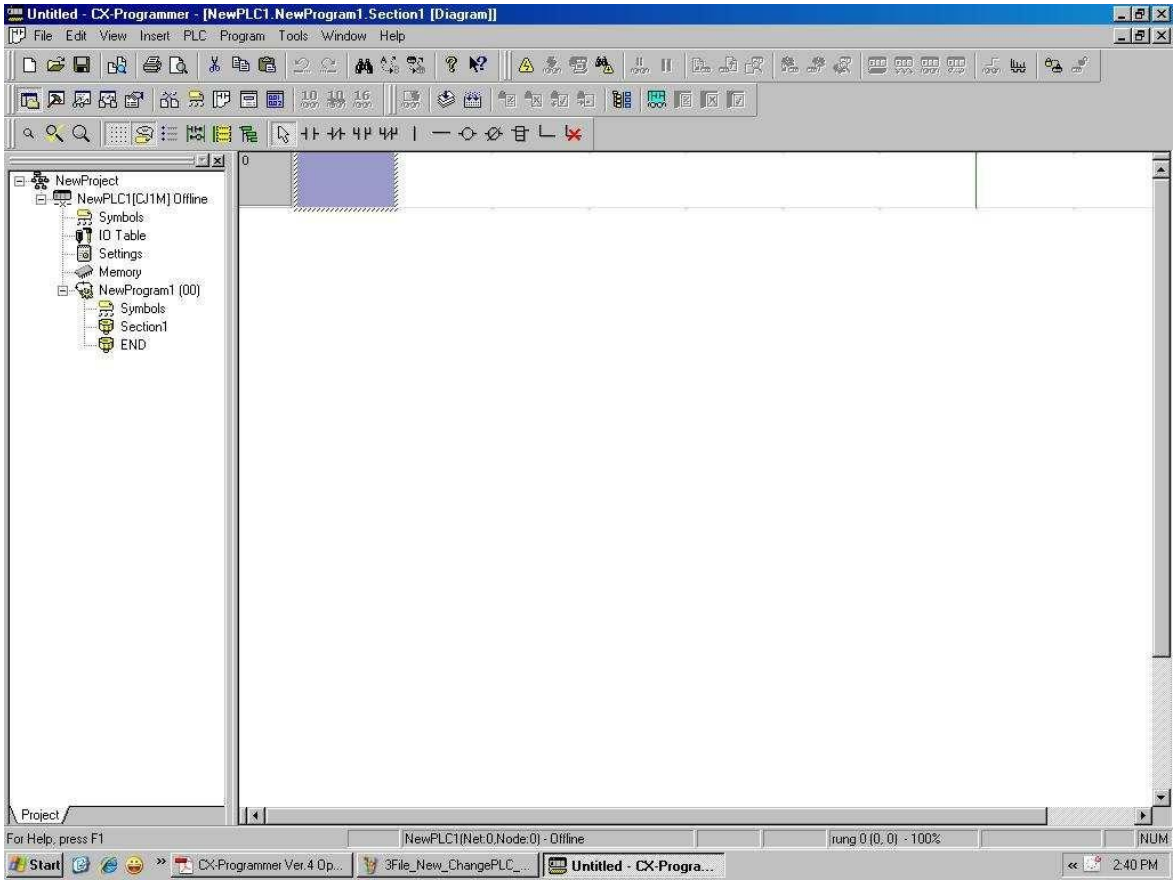


Pada bagian **Change PLC**, sub-bagian **Network Type**, pilih **SYSMAC-WAY**, kemudian pada kotak *Setting* lakukan pengaturan pada **COM** komputer yang digunakan.



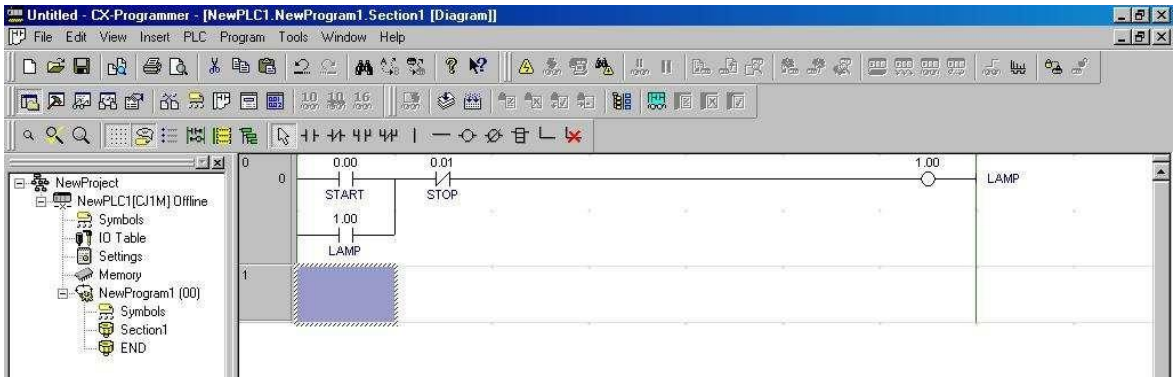
Setelah dipilih, tekan OK.

Akan muncul jendela pembuatan program pertama kali seperti gambar berikut. Setelah muncul jendela ini, maka komputer siap digunakan untuk membuat program.



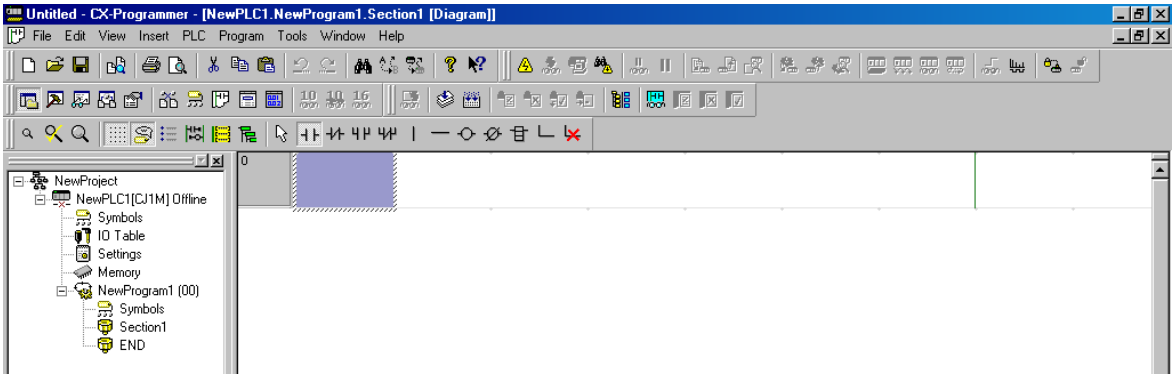
4.3 Contoh Pembuatan Program

Misal akan dibuat program berikut:



Terdapat dua *Input*, yaitu *Input Start* dengan alamat 0.00, dan *Input Stop* dengan alamat 0.01, serta sebuah *Output* Lampu dengan alamat 1.00.

Dengan menggunakan *mouse* pilih lambang **New Contact**, atau dengan menggunakan *keyboard* tekan huruf **C**.

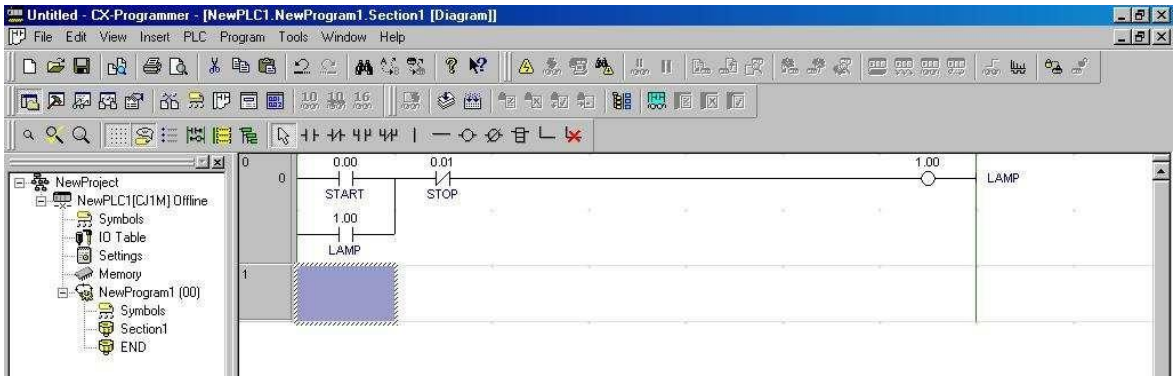


Isikan alamat dengan menuliskan alamat untuk *New Contact* tersebut (000), kemudian tekan enter. Kemudian pada **Edit Comment** ganti dengan ***Start***, setelah itu tekan enter.

Alamat 000 secara otomatis akan ditampilkan menjadi 0.00 pada *CX-Programmer*, sedangkan alamat 1.15 ditampilkan tetap. Semua proses yang sama dilakukan pula pada *Input* yang kedua yaitu **New Closed Contact (/)** dan pada *Output* yaitu **New Coil (O)**.

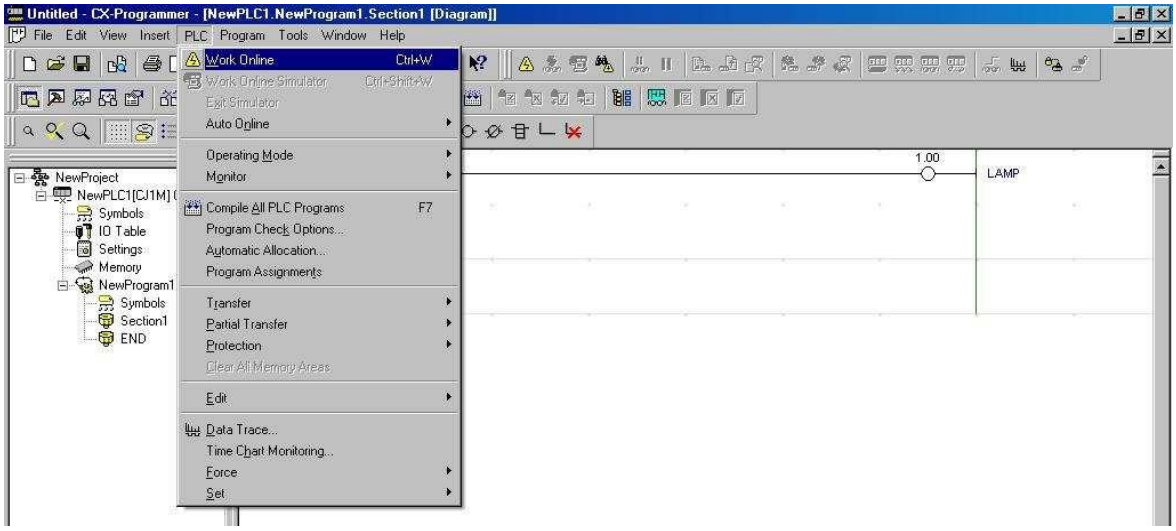
Penyambungan perintah yang ada menggunakan **Ctrl+anak panah**, demikian juga untuk memutuskan sambungan tersebut.

Setelah selesai maka tampilan akan menjadi seperti berikut:

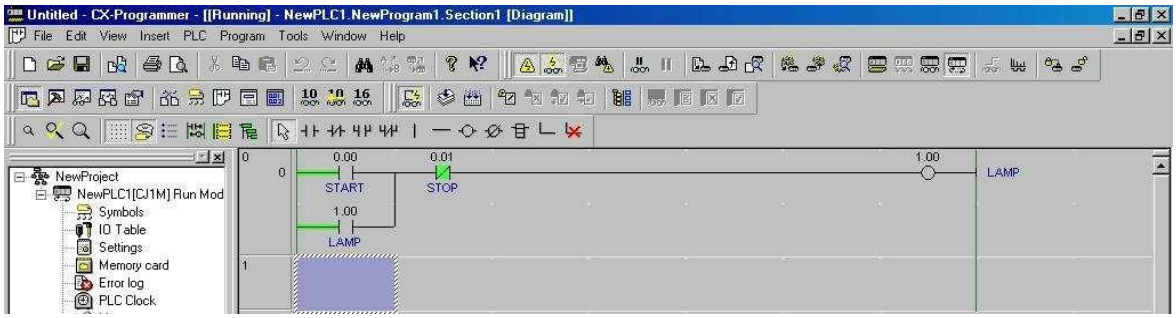


4.4 Men-*transfer* dan Menjalankan Program

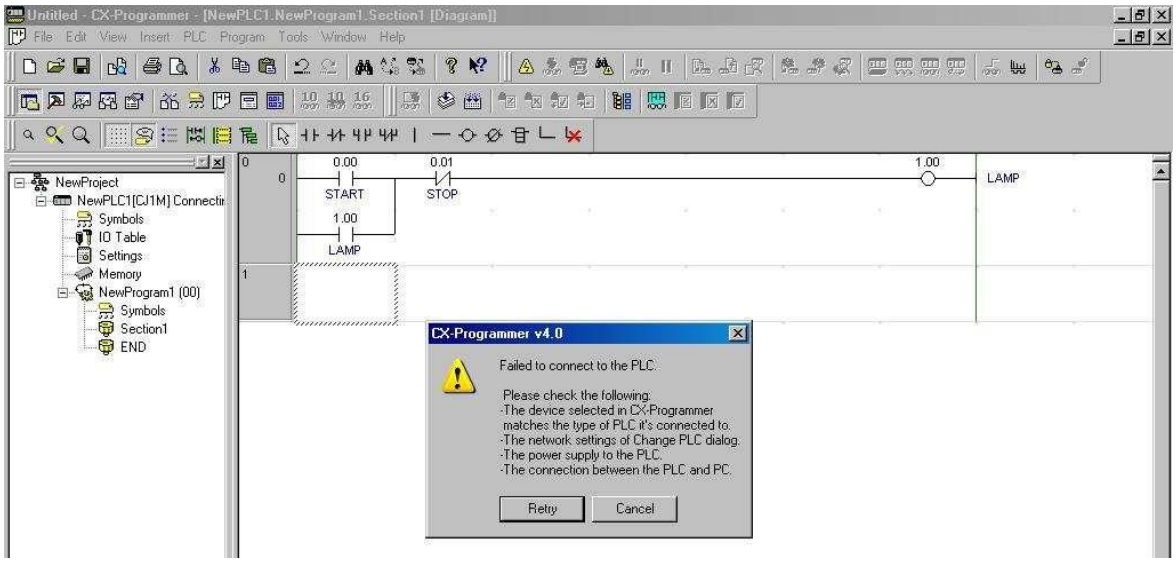
Program yang selesai dibuat, harus di-*transfer* dulu ke dalam memori program PLC. Sebelum di-*transfer*, PLC harus dihubungkan dulu dengan komputer. Cara menghubungkannya adalah dengan menekan **PLC-Work *Online***, akan muncul jendela berikut:



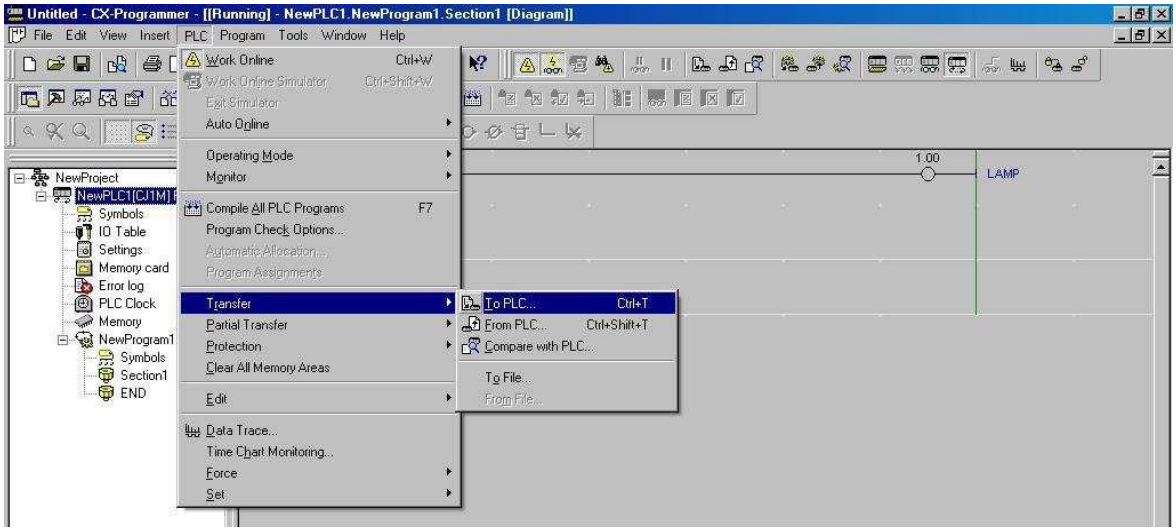
Jika tidak ada masalah pada cara koneksi, maka akan muncul jendela berikut:



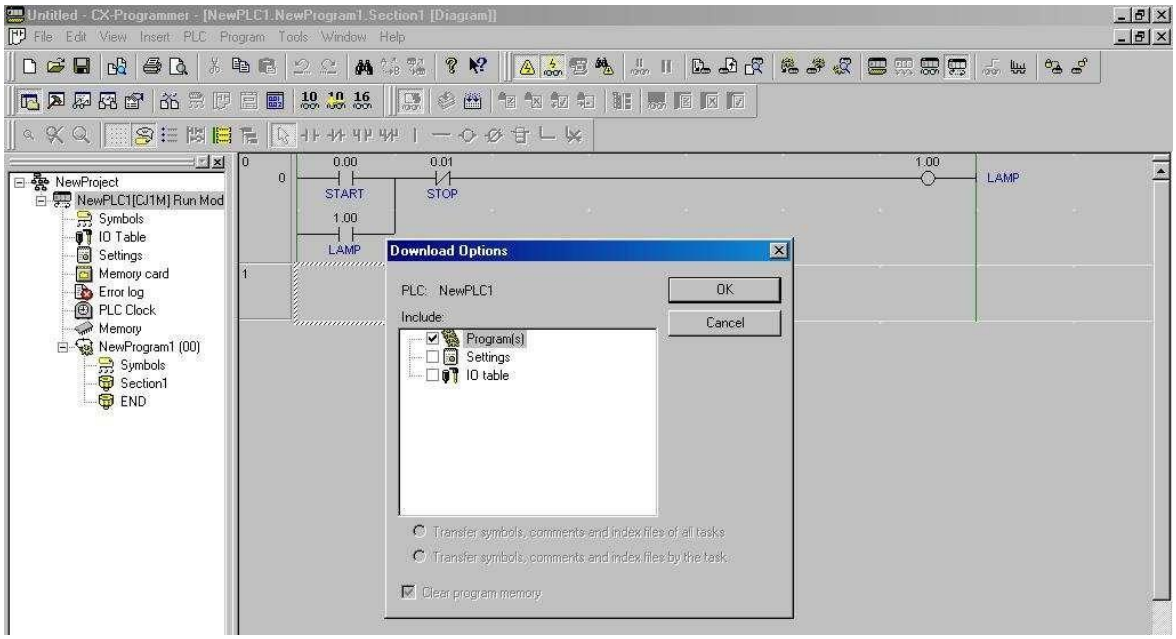
Jika ada masalah dengan koneksi antara PLC dan komputer, maka *CX-Programmer* akan memberi informasi berikut:



Jika muncul jendela ini, harus diperiksa apakah PLC sudah terhubung dengan sumber listrik, atau apakah koneksi kabel RS-232 sudah betul, atau pemilihan jenis PLC dengan PLC yang terhubung dengan komputer sudah sesuai dengan PLC yang digunakan.
Jika koneksi sudah benar, maka program yang dibuat dapat di-*transfer* ke dalam memori program PLC, caranya adalah tekan **PLC – Transfer - To PLC**.

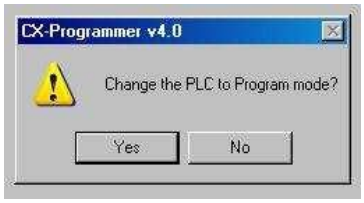


Ada pilihan yang akan di-*transfer*, tapi yang dipilih hanya **Program(s)** saja.

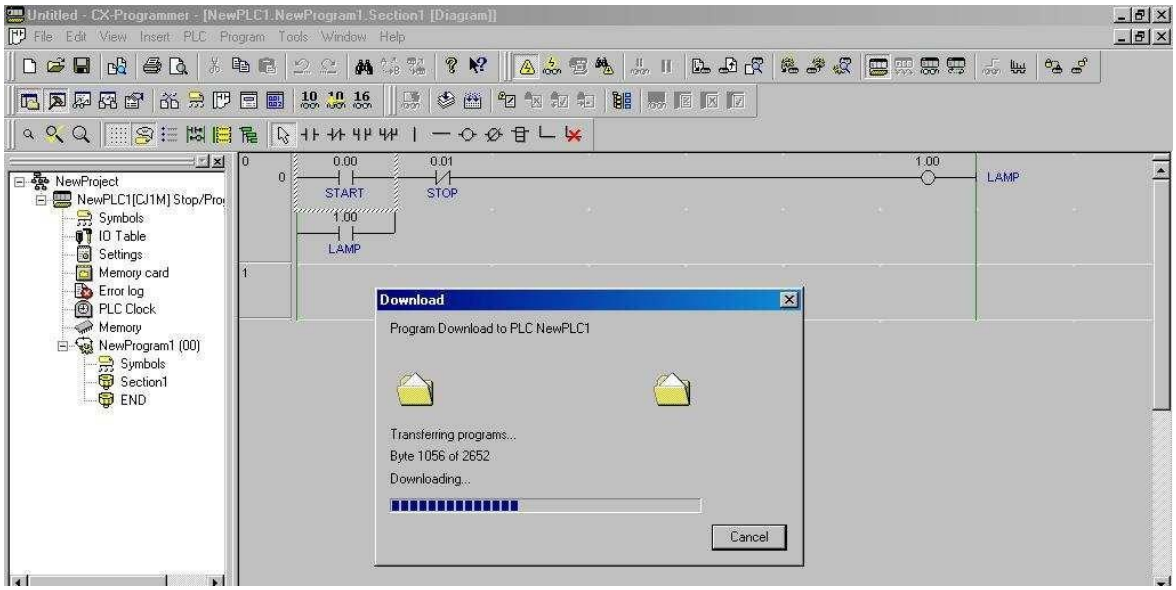


Setelah proses *transfer* selesai, tekan OK
Akan muncul pertanyaan ***This command will affect the state of the connected PLC***. Pilih Yes. Pilihan ini akan muncul jika sebelumnya PLC dalam kondisi RUN.

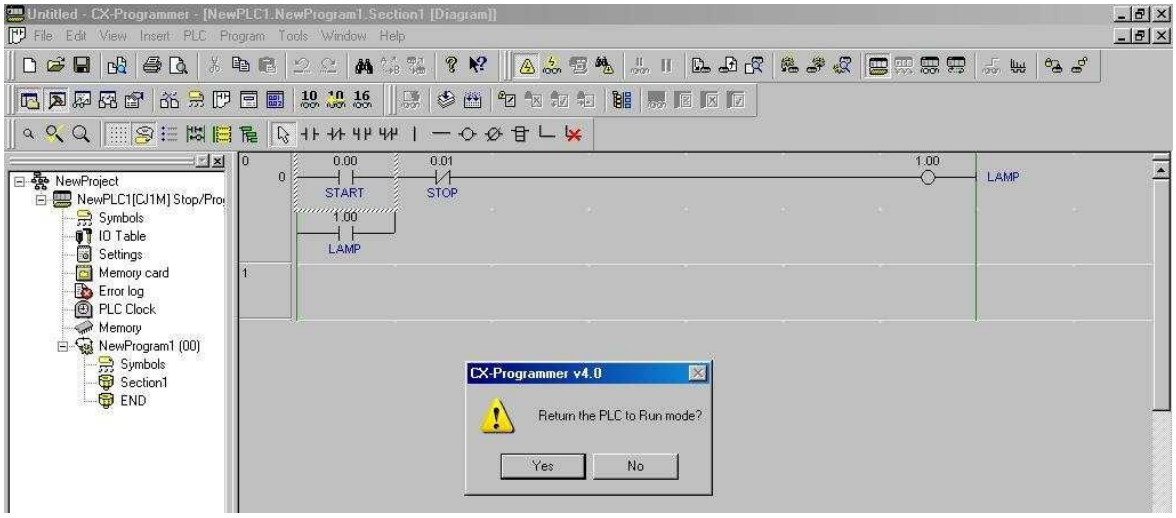
Kemudian *CX-Programmer* akan memberi pilihan ***Change the PLC to Program Mode?*** Pilih Yes.



Setelah dipilih Yes, maka *CX-Programmer* akan melakukan proses *transfer* program yang baru saja dibuat. Setelah selesai tekan OK.



Jika ada permintaan untuk kembali ke RUN Mode, tekan Yes.



Secara otomatis *CX-Programmer* akan membuat **Operating Mode** adalah **Run**, tapi jika ingin di *setting* manual, maka dapat dipilih dengan cara tekan **PLC - Operating Mode -** (pilih yang diinginkan). Program dapat dijalankan dengan mengaktifkan *Input Start* maka *Output* Lamp akan menyala, *Input Stop* diaktifkan maka *Output* Lamp akan mati.

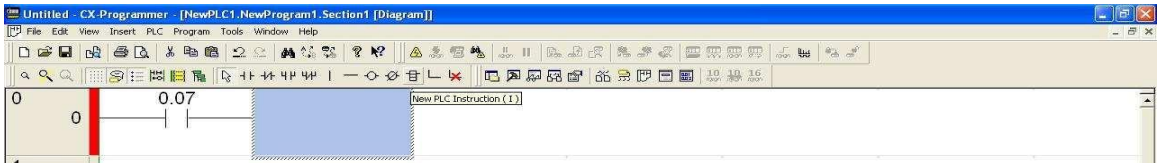
Contoh Program

Buat beberapa contoh program berikut, dan amati apa yang terjadi.

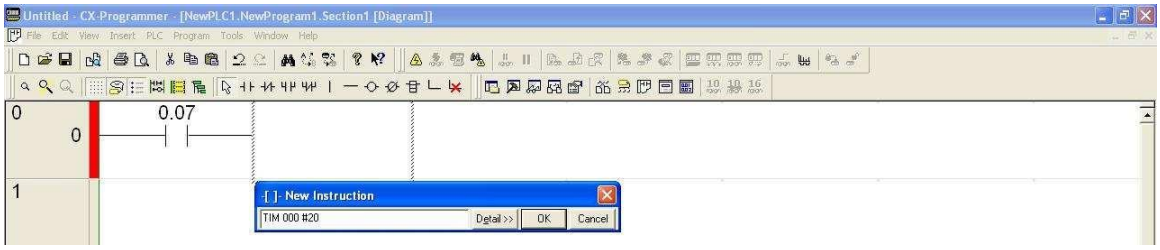
1. *Control Bit*

2. *Timer*

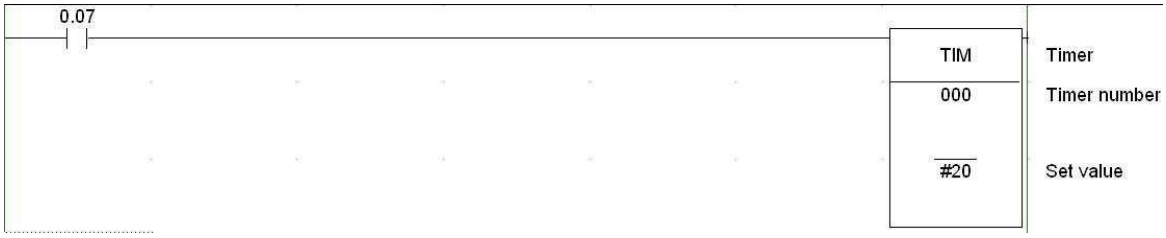
Cara membuat sebuah fungsi adalah dengan menekan menu **New PLC Instruction**, atau dengan cara lain yaitu menekan huruf **I** di sebelah kanan *input* yang telah dibuat (pada contoh gambar berikut setelah *input* beralamat 0.07).



Akan muncul perintah pengisian yang harus dilakukan untuk mengisi instruksi tersebut, jika *Timer* yang akan menjadi fungsi maka pengisiannya adalah ketik TIM 000 #20, TIM adalah nama fungsi, 000 adalah alamat *Timer*, dan #20 adalah *set value Timer*.

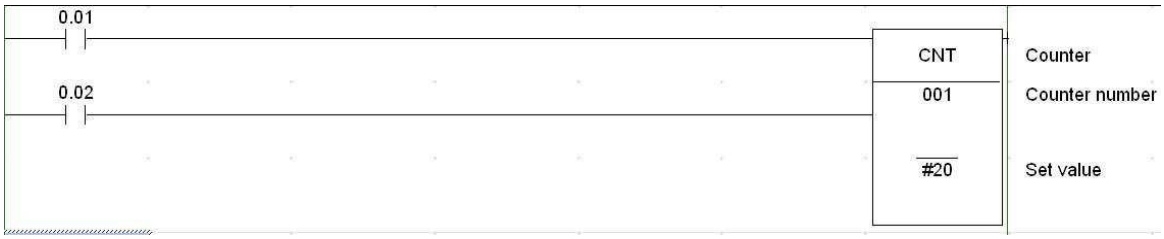


Jika pengisian sudah selesai, tekan enter. Hasil pengisian akan nampak seperti berikut:



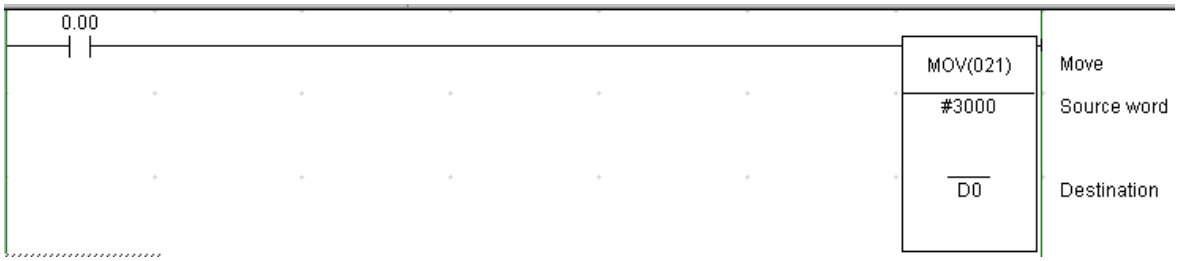
3. *Counter*

Untuk *Counter*, hampir sama dengan fungsi *Timer*, tetapi pengisian untuk *Counter* adalah seperti berikut: CNT 001 #20. CNT adalah fungsi *Counter*, 001 adalah alamat *Counter*, dan #20 adalah *preset value Counter*. Jika pengisian sudah benar maka akan nampak seperti berikut:



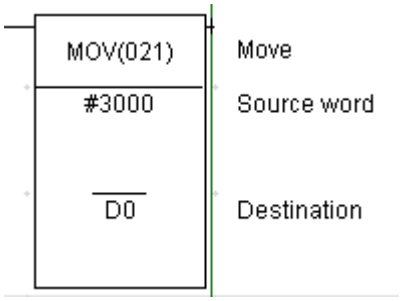
4. Instruksi lain

Untuk instruksi-instruksi lain, seperti KEEP, SET, +B, INC, MOV, dan sebagainya, pembuatannya hampir sama dengan perintah *Timer* atau *Counter*. Sebagai Contoh akan dibuat instruksi MOV(021) yang akan memindah data berupa data BCD #3000 ke alamat D0.



Tekan **I**, akan muncul instruksi **New Instruction**. Kemudian pada bagian isian ketik perintah mov #3000 d0.

Secara otomatis *CX-Programmer* akan mengubah perintah tersebut menjadi sebuah instruksi berikut:



Hal yang sama berlaku untuk instruksi-instruksi yang lain.

Soal-soal latihan:

Kerjakan soal-soal latihan berikut secara berurut, kuasai dengan baik penyelesaian satu soal baru kerjakan soal berikutnya.

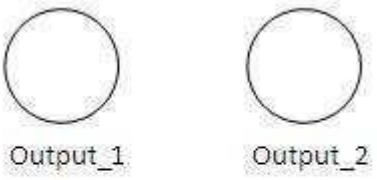
Self holding/Latching

1. Buatlah program berikut ini:

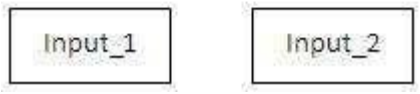
Amati apa yang terjadi jika:

- *Input pushbutton* 0.00 diaktifkan
- *Input pushbutton* 0.15 diaktifkan
- Kedua *input* tersebut diaktifkan bersamaan
- Buat program di atas dengan menggunakan instruksi KEEP(011)

2. Buatlah program untuk soal berikut:
Jika *Input*_1 di-*on*-kan maka *output*_1 akan *on*, demikian sebaliknya jika *Input*_2 di-*on*-kan maka *output*_2 akan *on*. Kedua *output* tidak boleh *on* bersamaan.



Input	*Register*
*Input*_1	0.00
*Input*_2	0.01

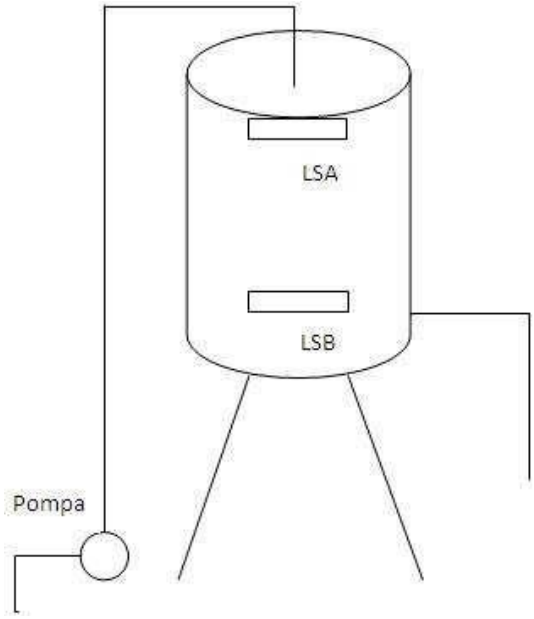


Output	*Register*
*Output*_1	1.05
*Output*_2	1.06

Amati apa yang terjadi jika:

- Kedua *Input* diaktifkan bersamaan
- Buat program di atas dengan menggunakan instruksi KEEP(011)

3. Buatlah program untuk soal berikut:



Sebuah instalasi tangki penyimpan air sederhana seperti pada gambar berikut. Asumsi awal tangki dalam keadaan kosong. Jika tangki kosong maka pompa akan bekerja memompa air dari sumur ke dalam tangki. Jika LSB (*Level Switch* Bawah) terkena bahkan terendam air, pompa masih bekerja. Pompa akan mati jika LSA (*Level Switch* Atas) tersentuh air. Karena air digunakan terus, maka level air akan turun. Karena level air turun maka ketika LSB tidak terendam air pompa akan bekerja kembali.

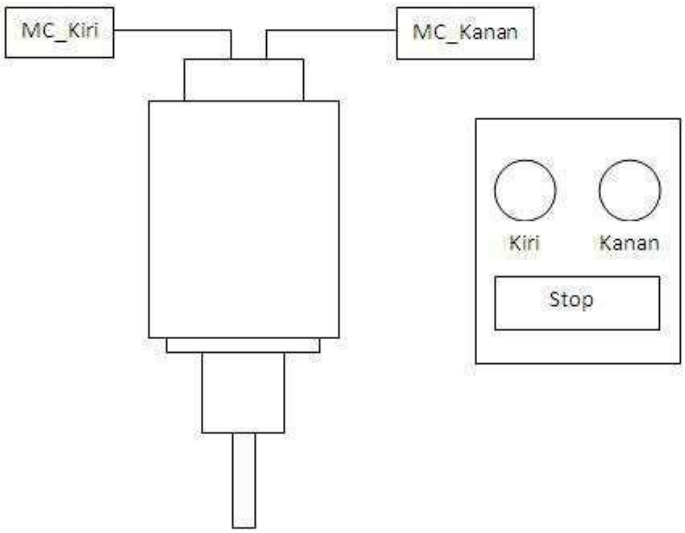
Input	*Register*
LSA	0.01
LSB	0.00

Output	*Register*
Pompa	1.05

Amati apa yang terjadi jika:

- Pada saat pompa bekerja dan level air berada di tengah kemudian listrik mati, apakah pompa akan terus bekerja atau pompa mati?
- Pada saat pompa tidak bekerja dan level air berada di tengah kemudian istrik mati, apakah pompa akan terus bekerja atau pompa mati?
- Buat program di atas dengan menggunakan instruksi KEEP(011)

4. Buatlah program untuk soal berikut:



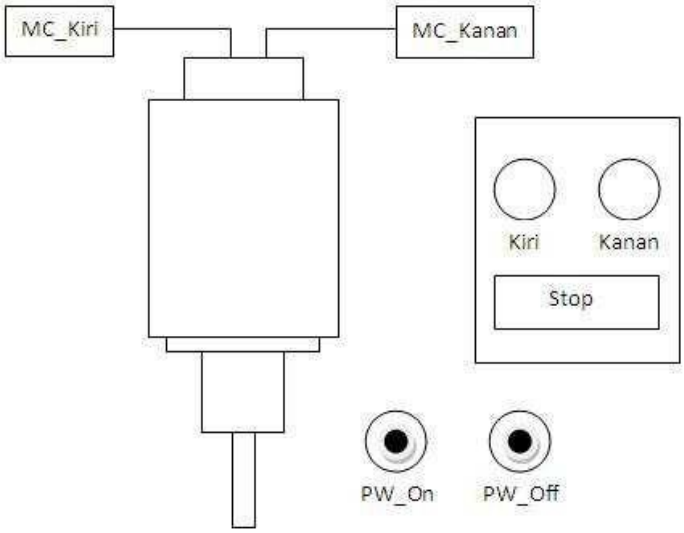
Terdapat sebuah mesin *milling* sederhana, ketika *Input* kiri di-*on*-kan maka MC_kiri (*Magnetic Contactor*_kiri) akan *on*. Saat MC_kiri *on*, *input* kanan di-*on*-kan tidak akan mengakibatkan MC_kanan *on*. Untuk meng-*on*-kan MC_kanan, tombol *stop* harus di-*on*-kan lebih dulu. Demikian juga sebaliknya.

Input	*Register*
Kiri	0.04
Stop	0.05
Kanan	0.06

Output	*Register*
MC_Kiri	1.04
MC_Kanan	1.06

Amati apa yang terjadi jika kedua *Input* kiri dan kanan di-*on*-kan bersamaan?

5. Buatlah program untuk soal berikut:



Sama dengan soal sebelumnya, hanya saja ditambahi dengan *Input* PW_*ON* (*Power_ON*) dan PW_*OFF* (*Power_OFF*) *Input PW_ON* berfungsi sebagai saklar *ON*. Jika saklar ini belum di-*on*-kan maka mesin *milling* ini tidak bisa dioperasikan

Input	*Register*
PW_*ON*	0.00
PW_*OFF*	0.01
Kiri	0.04
Stop	0.05
Kanan	0.06

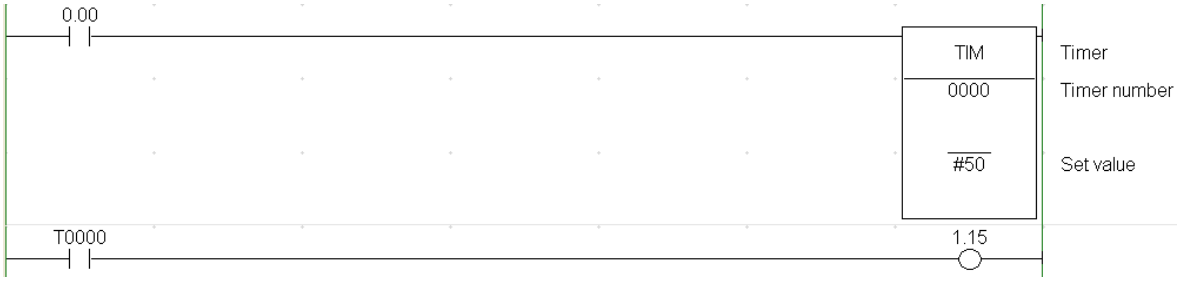
Output	*Register*
MC_Kiri	1.04
MC_Kanan	1.06

Amati apa yang terjadi jika:

- *Input* kiri atau *Input* kanan di-*on*-kan tetapi *Input* PW_*ON* belum di-*on*-kan
- *Output* MC_kiri atau MC_kanan sedang *on*, *Input* PW_*OFF* di-*on*-kan

Timer

1. Buatlah program berikut ini:



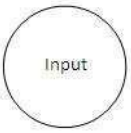
Aktifkan *input* 0.00, amati apa yang terjadi jika:

a
- SV *timer* mencapai 0?
- SV *timer* belum mencapai 0 tetapi *Input* 0.00 di-*off*-kan

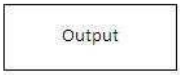
b
- *Input* 0.00 dibuat NC (*Normally closed*)

c
- *Input* 0.00 dan *Input* TIM000 dibuat NC

2. Buatlah program untuk soal berikut:



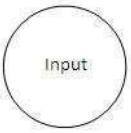
Jika *Input* di-*on*-kan maka *output* akan *on* selama 5 detik, setelah itu *output* akan mati sendiri.



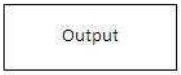
Input	*Register*
Input	0.00

Output	*Register*
Output	1.05

3. Buatlah program untuk soal berikut:



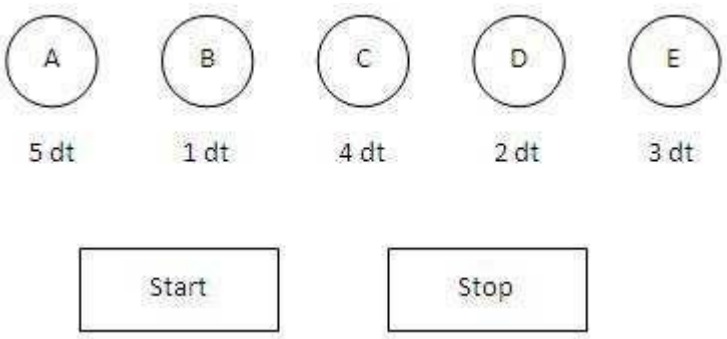
Jika *Input* di-*on*-kan maka *output* akan menunggu selama 5 detik baru akan *on*. *Output on* selama 3 detik, setelah itu *output* akan mati sendiri. Gunakan memori *internal* 200.00 untuk membantu membuat program ini.



Input	*Register*
Input	0.00

Output	*Register*
Output	1.05

4. Buatlah program untuk soal berikut:

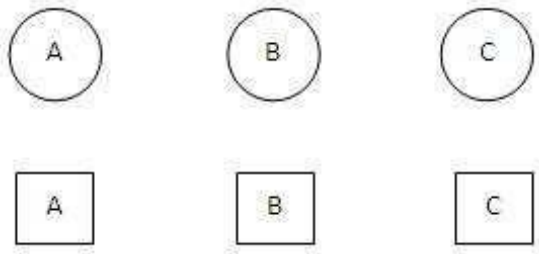


Terdapat dua *Input* yaitu *start* dan *stop*, serta 5 *output* yaitu *output* A, B, C, D, dan E. Jika *Input start* di-*on*-kan maka *output* A akan langsung *on* selama 5 detik. Setelah 5 detik *output* A akan *off*, kemudian *output* B akan *on* selama 1 detik. Demikian seterusnya sampai *output* E. Setelah *output* E mati, *output* A akan *on* lagi selama 5 detik. Proses tersebut akan terus berulang sampai *Input stop* diaktifkan.

Input	*Register*
Start	0.00
Stop	0.01

Output	*Register*
A	1.00
B	1.01
C	1.02
D	1.03
E	1.04

5. Buatlah program untuk soal berkut:



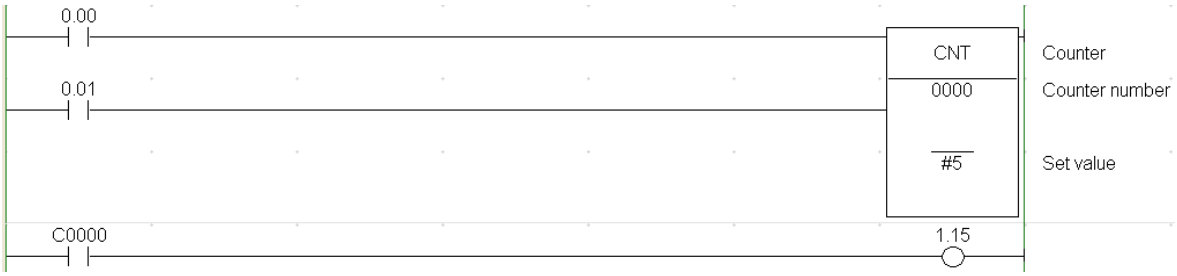
Terdapat 3 peserta kuis, yaitu A, B, dan C. Di masing-masing meja peserta terdapat tombol dan lampu. Jika juri membacakan pertanyaan dan peserta yang akan menjawab pertanyaan tersebut harus menekan tombol di meja peserta sendiri-sendiri. Siapapun yang menekan tombol lebih dulu lampu peserta tersebut akan menyala selama 5 detik dan lampu peserta lain tidak bisa menyala ketika salah seorang peserta sudah menyalakan lampunya.

Input	*Register*
A	0.04
B	0.05
C	0.06

Output	*Register*
A	1.04
B	1.05
C	1.06

Counter

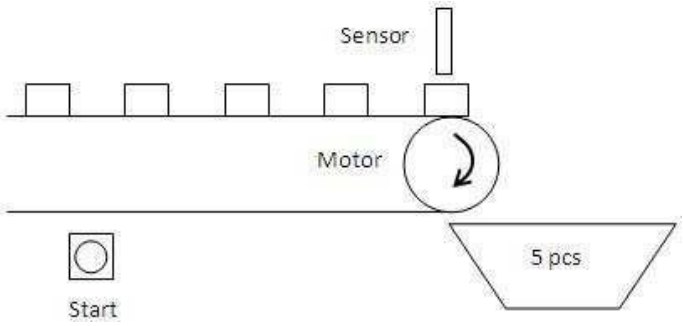
1. Buatlah program berikut ini:



Aktifkan *input* 0.00 beberapa kali, amati apa yang terjadi jika:

- PV *Counter* mencapai 0?
- PV *Counter* mencapai 0 dan *Input* 0.00 diaktifkan terus
- *Input* 0.01 di-*on*-kan jika PV *Counter* mencapai 0
- *Input* 0.01 di-*on*-kan jika PV *Counter* belum mencapai 0
- *Input* 0.01 dijadikan NC (*Normally closed*)

2. Buatlah program untuk soal berikut:

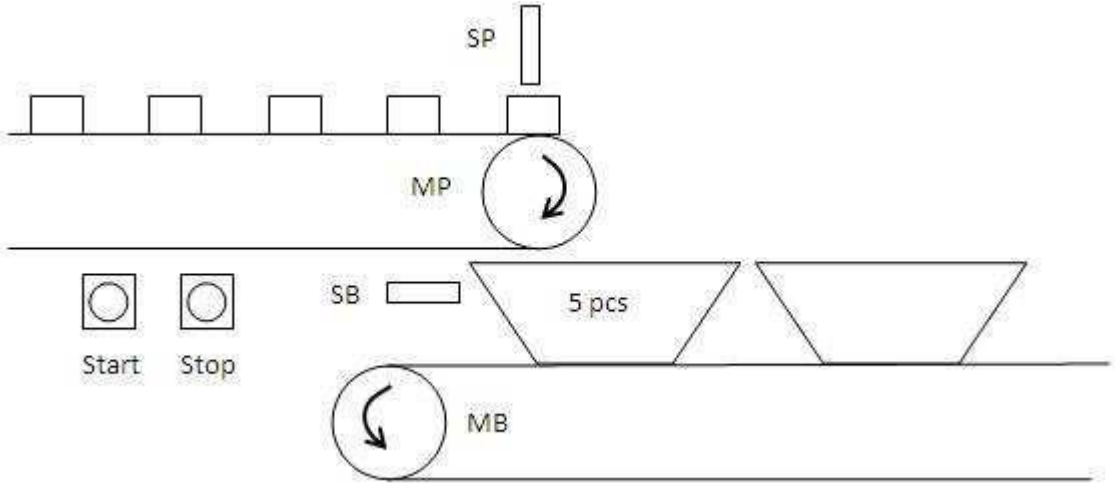


Terdapat suatu konveyor sederhana, jika *Input start* di-*on*-kan maka motor konveyor akan *on*. Jika motor konveyor *on* maka sensor akan mendeteksi produk yang melewatinya. Jika boks sudah terisi sampai 5 produk maka motor konveyor akan *off*. Untuk meng-*on*-kan motor konveyor *Input start* harus di-*on*-kan lagi.

Input	*Register*
Start	0.04
Sensor	0.05

Output	*Register*
Motor	1.04

3. Buatlah program untuk soal berikut:
   Terdapat dua konveyor yang bekerja, yaitu konveyor boks dan konveyor produk. Jika *Input start* di-*on*-kan maka konveyor boks akan berjalan, dan berhenti ketika SB (Sensor Boks) mendeteksi adanya boks.
   Begitu konveyor boks berhenti maka konveyor produk akan jalan dan SP (Sensor Produk) akan mendeteksi produk yang melewatinya. Jika sudah 10 produk yang berada dalam boks, konveyor produk akan berhenti dan konveyor boks akan berjalan lagi



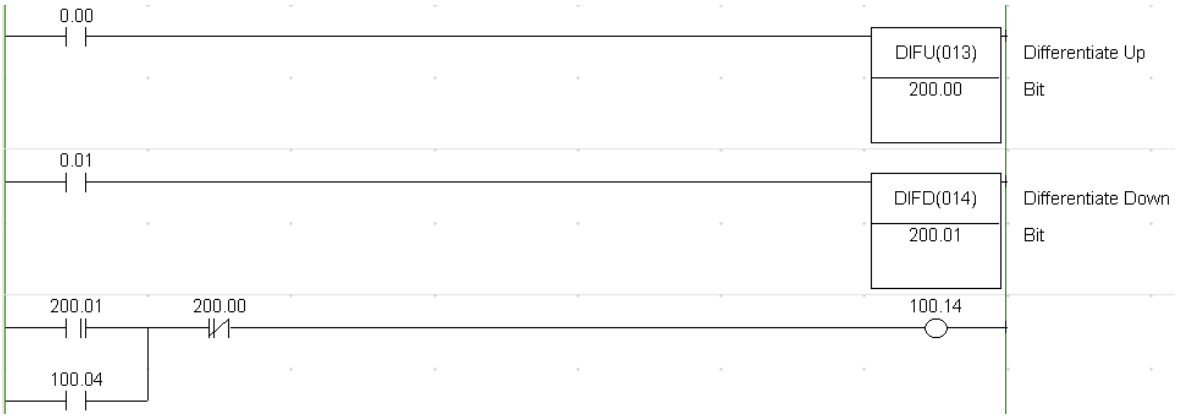
   tanpa harus meng-*on*-kan *Input start*. Proses akan berlangsung terus sampai *Input stop* diaktifkan.

Input	*Register*
Start	0.00
Stop	0.01
SB	0.02
SP	0.03

Output	*Register*
MB	1.04
MP	1.05

DIFU(013) dan **DIFD(014)**

1. Buatlah Contoh program berikut:

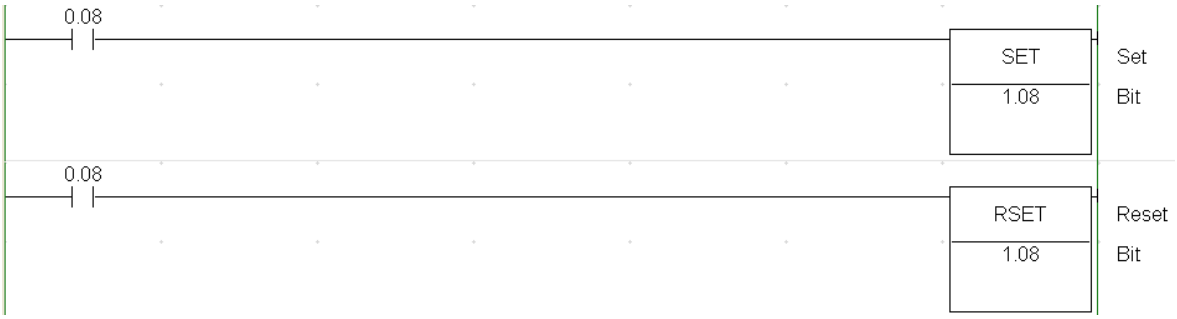


- Amati apa yang terjadi jika *input* 0.00 diaktifkan?
- Amati apa yang terjadi jika *input* 0.01 diaktifkan?
- Apa pengaruh status *input*, dari *off* menuju ke *on* atau sebaliknya terhadap *output*?

2. *Input* dari sensor ultrasonik (0.00) digunakan untuk mendeteksi kendaraan yang akan masuk, yang akan memerintahkan motor listrik (1.00) untuk membuka pintu. Sebuah sensor proksimiti (0.02) akan memerintahkan motor berhenti bekerja membuka pintu jika pintu sudah sampai atas. Sebuah sensor inframerah (0.01) akan memberi sinyal ke PLC jika kendaraan sudah melewati pintu, dan memerintahkan motor listrik (1.01) untuk menutup pintu, dan sebuah *limit switch* (0.04) akan tersentuh yang menyebabkan motor listrik berhenti bekerja. Buat programnya, dan amati apa yang terjadi jika:
   - Sebelum pintu menyentuh proksimiti, motor akan terus bekerja membuka atau menutup?
   - Ketika proses menutup, ada kendaraan lain yang akan masuk, motor listrik akan terus menutup pintu atau membuka pintu?

SET dan **RSET**

1. Buatlah program berikut, kemudian jalankan.



- Amati apa yang terjadi jika *input* 0.08 diaktifkan
- Amati apa yang terjadi jika *input* 0.09 diaktifkan
- Amati apa yang terjadi jika kedua *input* tersebut diaktifkan

2. Buat program pada soal nomor 1, 2, 3, dan 4 pada contoh *self holding* menggunakan instruksi SET dan RSET.

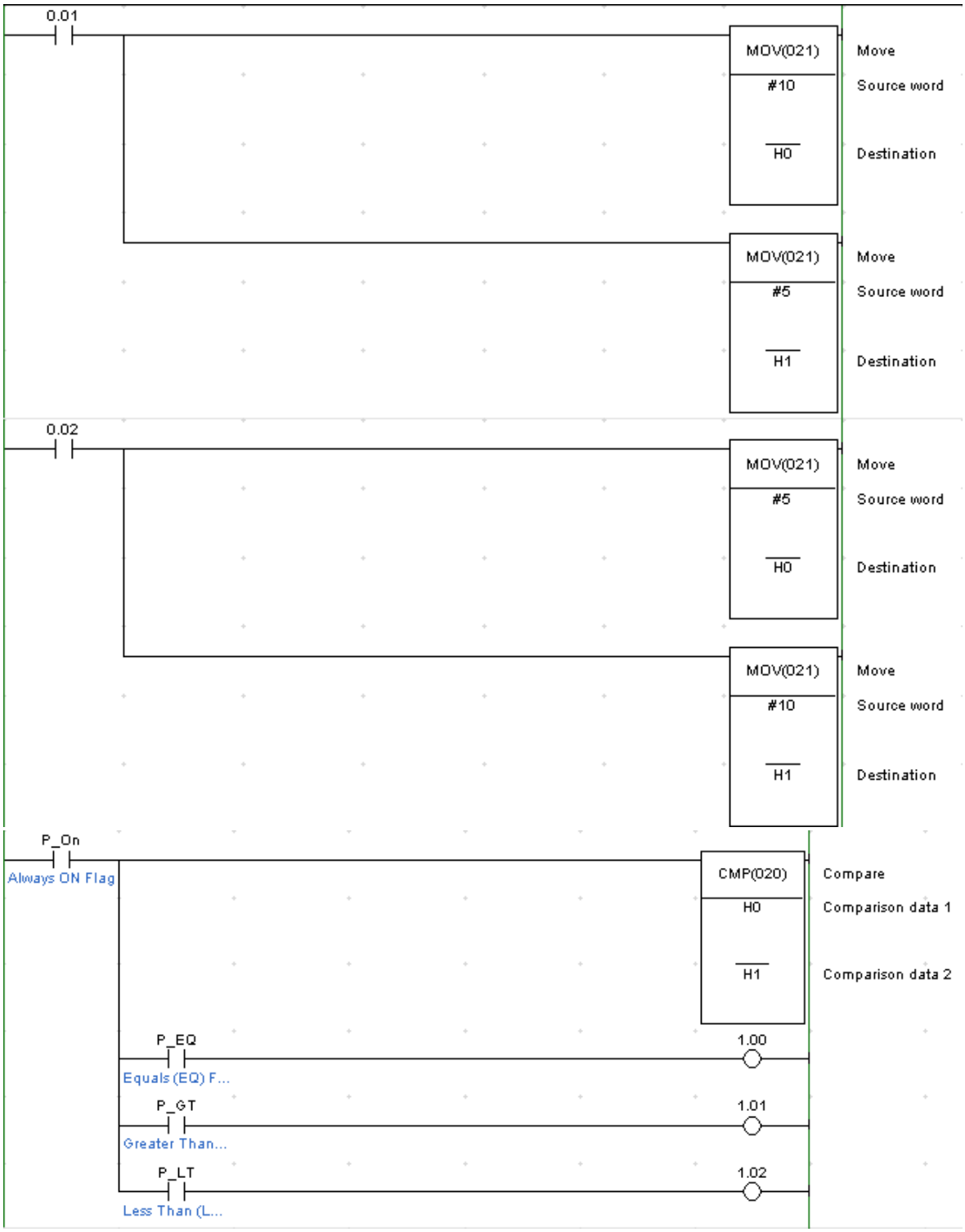
3. Buat program pada soal nomor 1, dan 2 pada contoh DIFU(013) dan DIFD(014) menggunakan instruksi SET dan RSET.

Instruksi MOV(021) dan **CMP(020)**

1. Buatlah program berikut, kemudian jalankan.

- Amati apa yang terjadi jika *input* 0.01 diaktifkan
- Amati apa yang terjadi jika *input* 0.02 diaktifkan

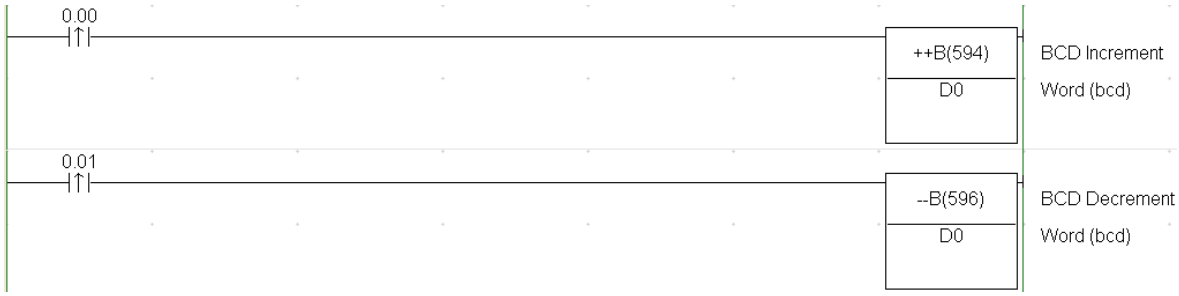
2. Buatlah program berikut, kemudian jalankan.



- Amati apa yang terjadi jika *input* 0.01 diaktifkan
- Amati apa yang terjadi jika *input* 0.02 diaktifkan

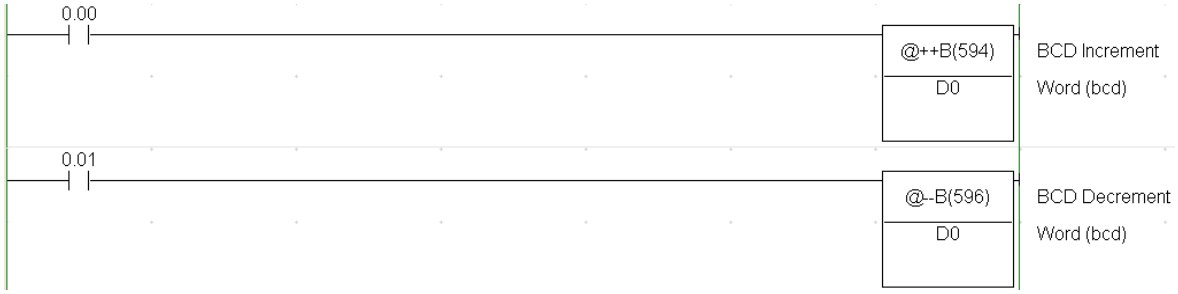
Instruksi INCREMENT BCD: ++B(594) dan **DECREMENT BCD: --B(596)**

1. Buatlah program berikut, kemudian jalankan.



- Amati apa yang terjadi jika *input* 0.00 diaktifkan
- Amati apa yang terjadi jika *input* 0.01 diaktifkan

2. Buatlah program berikut, kemudian jalankan.



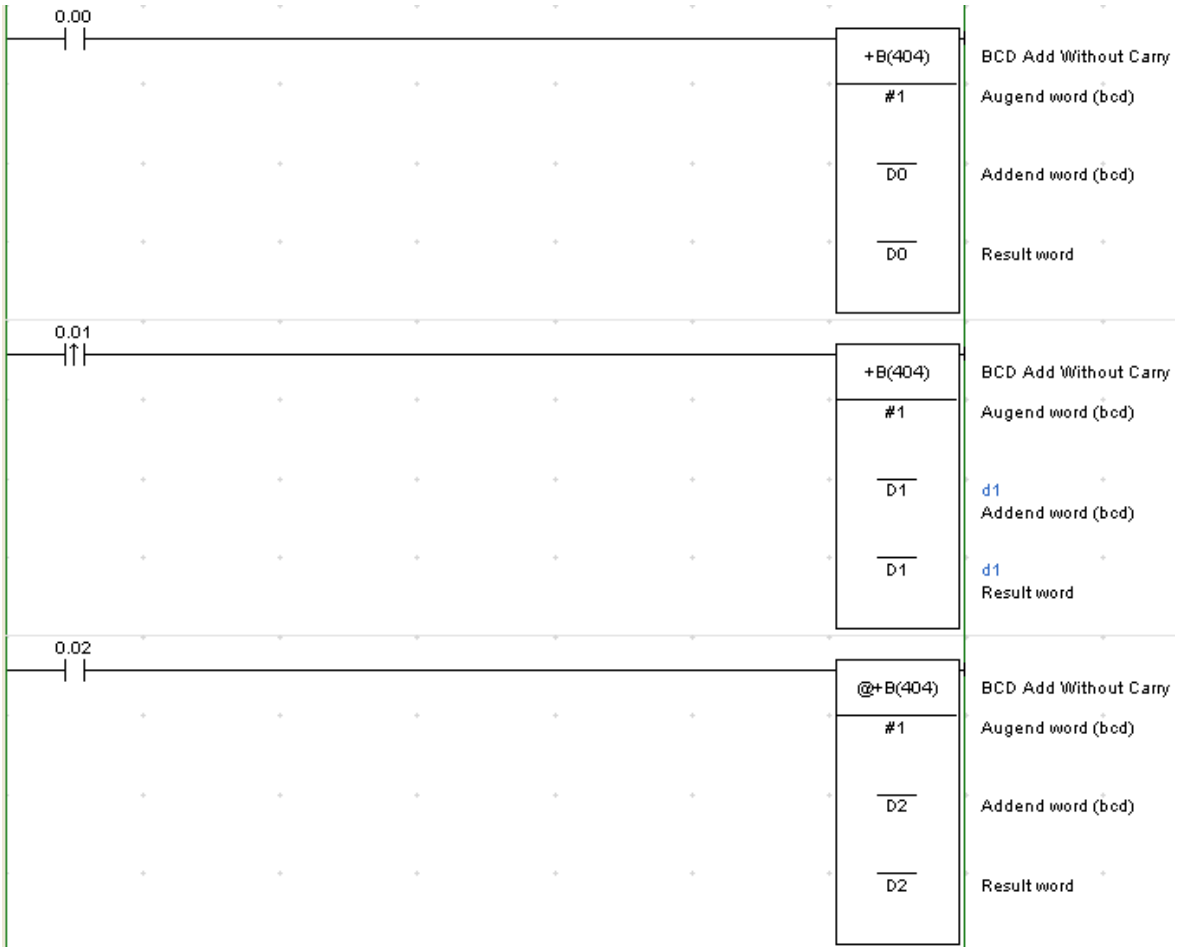
- Amati apa yang terjadi jika *input* 0.00 diaktifkan
- Amati apa yang terjadi jika *input* 0.01 diaktifkan

3. Muatlah program berikut, kemudian jalankan.

- Amati apa yang terjadi jika *input* 0.00 diaktifkan
- Amati apa yang terjadi jika *input* 0.01 diaktifkan
- Apa kesimpulan yang bisa anda ambil dari soal nomor 1, 2, dan 3?

Instruksi Aritmetika sederhana

1. Buatlah program berikut, kemudian jalankan.



- Amati apa yang terjadi jika *input* 0.00 diaktifkan
- Amati apa yang terjadi jika *input* 0.01 diaktifkan
- Amati apa yang terjadi jika *input* 0.02 diaktifkan
- Apa yang bisa anda simpulkan?

2. Buat program pada soal nomor 2 dan 3 mengenai *Counter* dengan menggunakan instruksi +B(404) dan CMP(020).

3. Buatlah program berikut, kemudian jalankan.

- Amati apa yang terjadi jika *input* 0.00 diaktifkan
- Amati apa yang terjadi jika *input* 0.01 diaktifkan
- Amati apa yang terjadi jika *input* 0.02 diaktifkan
- Apa yang bisa anda simpulkan?

4. Buat program pada soal nomor 2 dan 3 mengenai *Counter* dengan menggunakan instruksi +B(404) dan CMP(020).

5. Buat program untuk pengaturan lampu lalu lintas di sebuah perempatan menggunakan instruksi-instruksi yang sudah dipelajari. Lama waktu untuk lampu merah adalah 10 detik, lampu kuning 2 detik, dan lampu hijau 10 detik untuk masing-masing jalan.