SISTEM MIKROPROSESOR

PERTEMUAN 2

Mochamad Fajar Wicaksono, S.Kom., M.Kom.

BAHASA ASSEMBLY, PENGALAMATAN AVR DAN OPERASI PORT I/O

BAHASA ASSEMBLY

- Program-program bahasa rakitan ditulis dengan singkatan-singkatan yang pendek yang disebut mnemonic.

- Mnemonic singkatan yang merepresentasikan instruksi mesin aktual.

- Pemrograman bahasa rakitan penulisan instruksi-instruksi mesin dalam bentuk mnemonic, dimana setiap instruksi mesin (biner atau hex) digantikan oleh sebuah mnemonic.

BAHASA ASSEMBLY

- Bahasa assembly terdiri atas sebuah rangkaian assembly statement dimana statement ditulis satu per baris.
- Medan (field): label, opcode,operand dan komentar

[Label]	Opcode	[Operand]	[Komentar]
Awal:	ldi	R16,0x03	;isi r16 dengan data 03h

BAHASA ASSEMBLY

- **Label** : mewakili alamat instruksi (atau data). Label harus diakhiri dengan titik dua. Tidak boleh diawali angka.
- **Operand**
- **Komentar**: biasanya berisi penjelasan untuk baris program dan harus didahului tanda “;”
- Setiap peralihan dari satu medan alamat ke medan alamat berikutnya harus dipisahkan minimal satu spasi/blank atau dengan tombol tab.
- **Tanda []** menunjukkan medan tersebut sifatnya opsional.

BAHASA ASSEMBLY

- Pada AVR jumlah maksimum operand adalah dua operand.
- AVR mengenal 3 macam pengaksesan operand, yaitu nol operand, satu operand, dan dua operand.

Jenis instruksi	Instruksi	Komentar
Nol operand	CLC	Buat carry flag berlogika low
	SEC	Buat carry flag berlogika high
Satu operand	CLR r16	Buat register r16 berlogika low
	SER r16	Buat register r16 berlogika high
Dua operand	LDI r16,0x05	Isikan 25H ke register r16
	OUT PORTA,r16	Kirimkan byte di r16 ke Port A

CONTOH PENULISAN PROGRAM

```
.include “m8535.inc”
.org 0x0000
rjmp awal
awal:
LDI r16,low(ramend)
OUT SPL,r16
LDI r16,high(ramend)
OUT SPH,r16

LDI r16,0x25
LDI r17,0x19
ADD r16,r17
AKHIR: JMP AKHIR
```

Software Studio 5

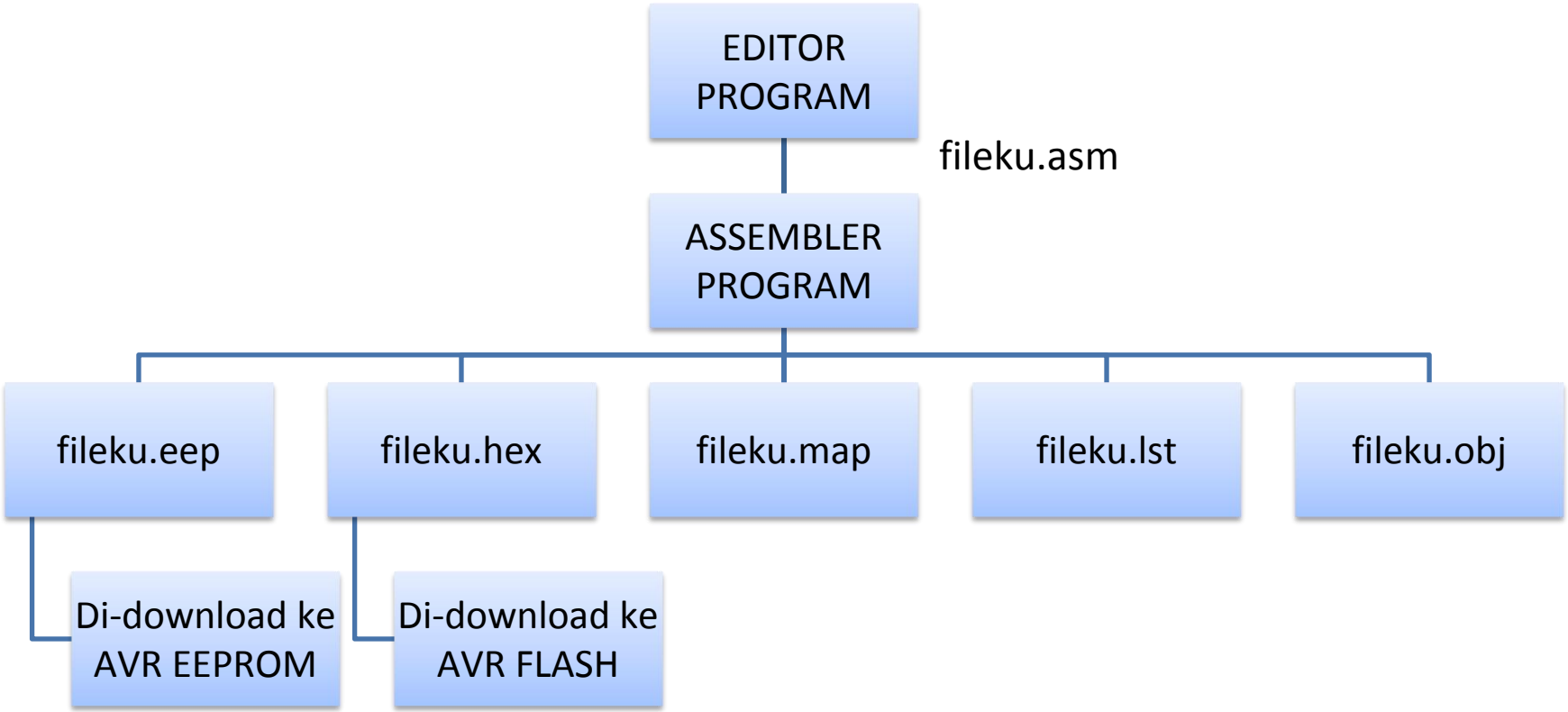
INISIALISASI PROGRAM

PROGRAM UTAMA

LANGKAH INISIALISASI PROGRAM

- Menentukan jenis mikrokontroller yang digunakan dengan cara memasukan file definisi kedalam program utama.
- Menuliskan awal alamat program, yaitu .org 0x0000. kemudian dilanjutkan dengan instruksi rjmp ke label AWAL. Hal ini dimaksudkan agar program memori tidak tumpang tindih dengan memori data.
- Menentukan isi Stack Pointer dengan alamat terakhir RAM (RAMEND). Hal ini dimaksudkan agar program mulai ditulis setelah alamat terakhir RAM.

PERAKITAN PROGRAM AVR



FORMAT DATA AVR
DAN ASSEMBLER DIRECTIVE

TIPE DATA AVR

- Mikokontroler AVR hanya memiliki satu tipe data, yaitu 8bit.
- Ukuran setiap register juga 8bit.

REPRESENTASI FORMAT DATA

- **Bilangan HEX**
  - Letakkan 0x didepan bilangan misal ldi r16,0x25
  - Letakkan $ didepan bilangan misal ldi r16,$25
- **Bilangan Biner** : letakkan 0b didepan bilangan biner misal ldi r16,0b00011001
- **Bilangan Desimal**: menuliskan bilangan tanpa ada tambahan apapun misal r16,19
- **Karakter ASCII**: gunakan tanda petik tunggal yang mengapit karakter ASCII misal ldi r16,’5’

ASSEMBLER DIRECTIVE

- Merupakan control instruksi pada program assembly tersebut, tetapi bukan sebagai Bahasa assembly yang akan dijalankan oleh mikrokontroler.
- Berfungsi mengubah penunjuk kode assembly.

ASSEMBLER DIRECTIVE

1. **.cseg (code segment)** pengarah ini berguna sebagai penunjuk bahwa kode atau ekspresi dibawahnya **diletakkan pada memori program** pengarah ini biasanya diletakkan setelah pengarah **.deseg**
2. **.db (data byte)** pengarah ini memungkinkan kita dapat meletakkan konstanta seperti serial number, dan **lookup table di memory program** pada alamat tertentu.
3. **.dw (data word)** pengarah ini seperti data byte, tetapi dalam ukuran word.
4. **.org** digunakan untuk mengeset program counter pada alamat tertentu.
5. **.byte** digunakan untuk inisialisasi besar byte yang digunakan pada SRAM untuk label tertentu
6. **.dseg (data segment)** pengarah ini berguna sebagai penunjuk bahwa kode dibawahnya **berfungsi untuk melakukan setting SRAM**
7. **.def (define)** pengarah ini memungkinkan suatu register dapat didefinisikan dengan nama lain. Contoh **def temp=r16 ; temp nama lain dari register r16**

ASSEMBLER DIRECTIVE

8. **.equ** berguna untuk memberi nama suatu konstanta yang tidak dapat berubah. Misal **.EQU counter = 0x19**
9. **.set** sama seperti **.equ** tetapi konstantanya dapat diubah. Misal **.set baud = 9600**
10. **.endm (end macro)** untuk mengakhiri macro.
11. **.include** untuk menggabungkan sebuah file kedalam program agar program lebih cepat dimengerti atau memisahkan kode dalam dua file terpisah.
12. **.device** sebagai penunjuk jenis AVR yang digunakan.
13. **.exit** sebagai penunjuk agar berhenti melakukan assembly pada file saat ini.
14. **.list** berguna membangkitkan file list.
15. **.listmac** berguna agar penambahan macro ditampilkan pada file list yang dibangkitkan.
16. **.nolist** berguna agar suatu runtun instruksi tidak dimasukkan dalam file list yang dibangkitkan.

GENERAL PURPOSE REGISTER (GPR)

TIPS Pemrograman dan Register-Register AVR

- Definisi file include

  Ditujukan agar kita dapat langsung mengalamati register-register yang dimiliki ATMEGA8535. (semua AVR memiliki 32 GPR yaitu R0-R31)

- Register

  Register r0 s/d r15 tidak dapat digunakan untuk pengalamatan immediate data.

GENERAL PURPOSE REGISTER (GPR)

Register

- Terdapat register khusus register X, register Y dan register Z untuk operasi-operasi 16 bit.
- Register ini digunakan untuk
  1. Membaca dan menulisi SRAM.
  2. Membaca memori program, misalnya pembacaan lookup table.

GENERAL PURPOSE REGISTER (GPR)

Contoh:

```
Ldi YH,high(LABEL)
Ldi YL,low(LABEL)
```

LABEL adalah alamat untuk lookup table atau lokasi memori

TIPS PENGGUNAAN REGISTER

- Tetapkan nama-nama register dengan pengarah **.DEF**
- Jika kita memerlukan akses pointer, maka cadangkanlah r26 s/d r31.
- Pencacah/counter 16bit paling baik ditempatkan di r25:24
- Jika kita ingin membaca dari memori program, misalnya fixed table, maka cadangkan Z (r30:r31) dan R0.
- Jika kita merencakanan harus mengakses bit-bit tunggal dalam suatu register (misalnya pengujian flag) gunakan r16 s/d r23.

MODE PENGALAMATAN AVR

1. Pengalamatan immediate
2. Pengalamatan register
3. Pengalamatan langsung
4. Pengalamatan tak-langsung
5. Pengalamatan tak-langsung index
6. Pengalamatan tak-langsung displacement.

Mode pengalamatan tak-langsung AVR mode pengalamatan tak-langsung register.

Instruksi Pembacaan Memori SRAM

Instruksi Pembacaan Memori SRAM

- Pembacaan data dari SRAM ke register file (general purpose register) menggunakan **instruksi load.**

Instruksi Pembacaan Memori SRAM

LDS Rn,Alamat_SRAM

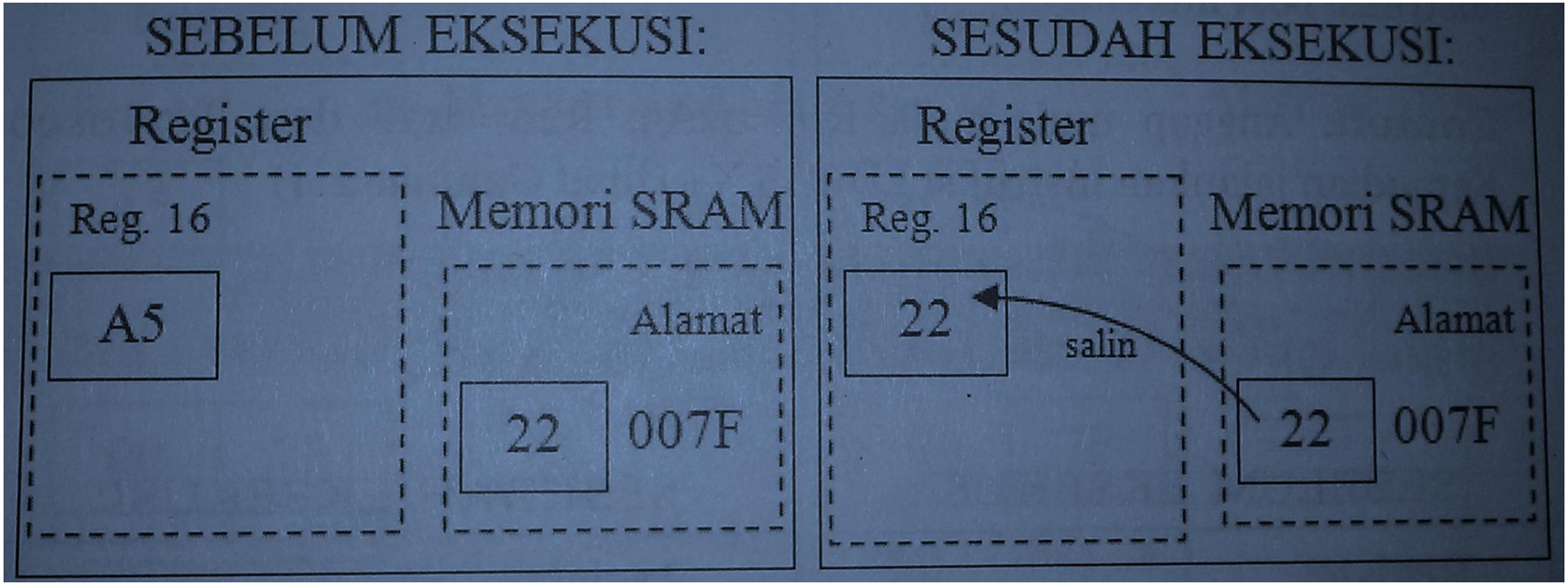
Load data di alamat SRAM ke register Rn. Alamat SRAM adalah immediate value.

- Contoh:
- R16 =0xa5 , lalu [0x007F]=0x22

- Lds r16,0x007F ; salin data dari alamat SRAM
  ;0x007F ke r16

ILUSTRASI

Lds r16,0x007F



Instruksi Pembacaan Memori SRAM

LD Rn,X/Y/Z

Instruksi ini akan me-load nilai yang tersimpan di lokasi memori yang ditunjuk oleh register X/Y/Z ke register tujuan Rn (bisa R0…R31).

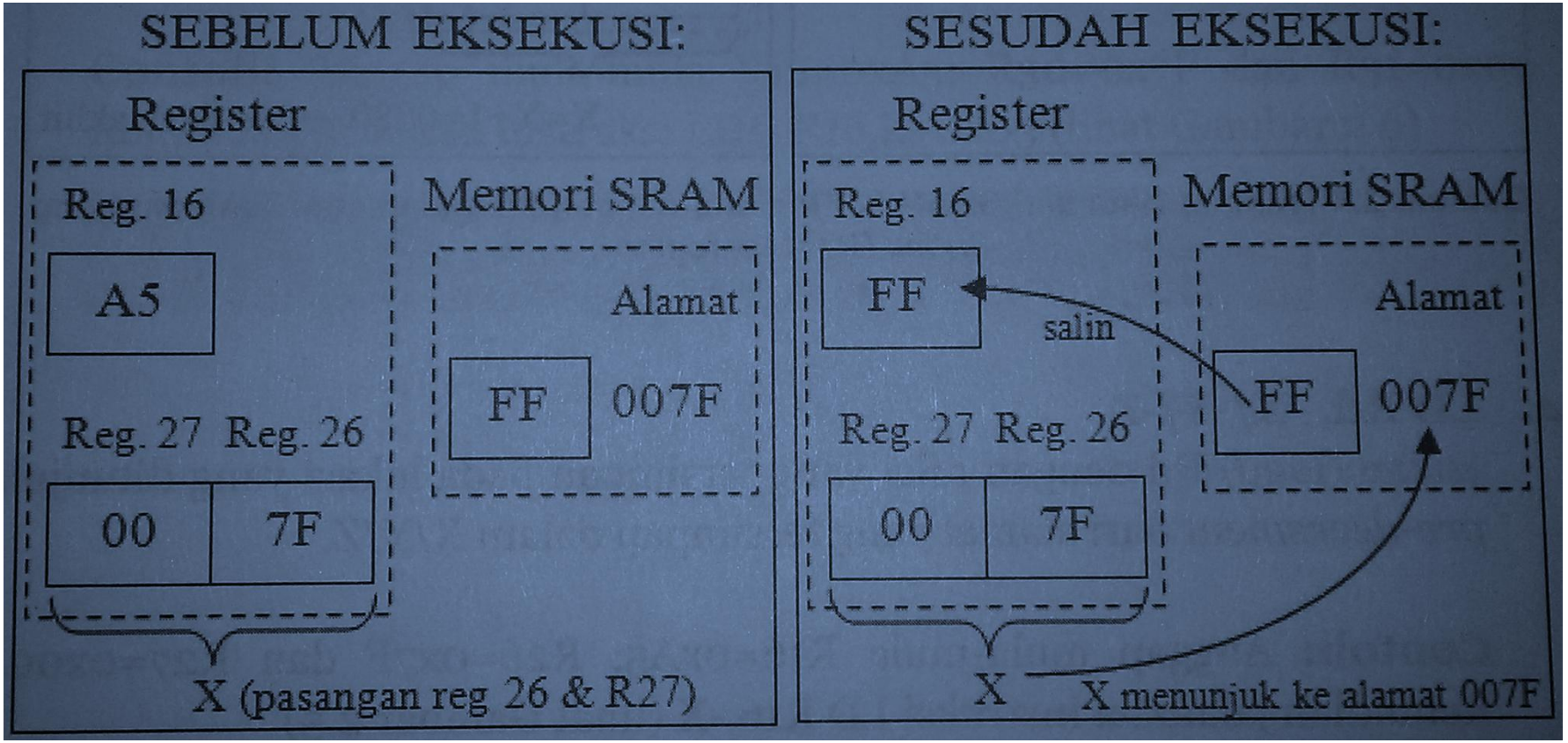
Misal:

R16=0xa5; **r26=0x7F** ; **r27=0x00**

Instruksi **LD r16,X**

ILUSTRASI

LD r16,X



Instruksi Pembacaan Memori SRAM

LD Rn,X+/Y+/Z+

Instruksi ini akan mengisi Rn dengan nilai yang tersimpan dimemori pada lokasi yang ditunjuk oleh register X/Y/Z dan kemudian menambah lokasi memori dengan 1 (increment). Instruksi ini adalah sebuah instruksi **post increment.**

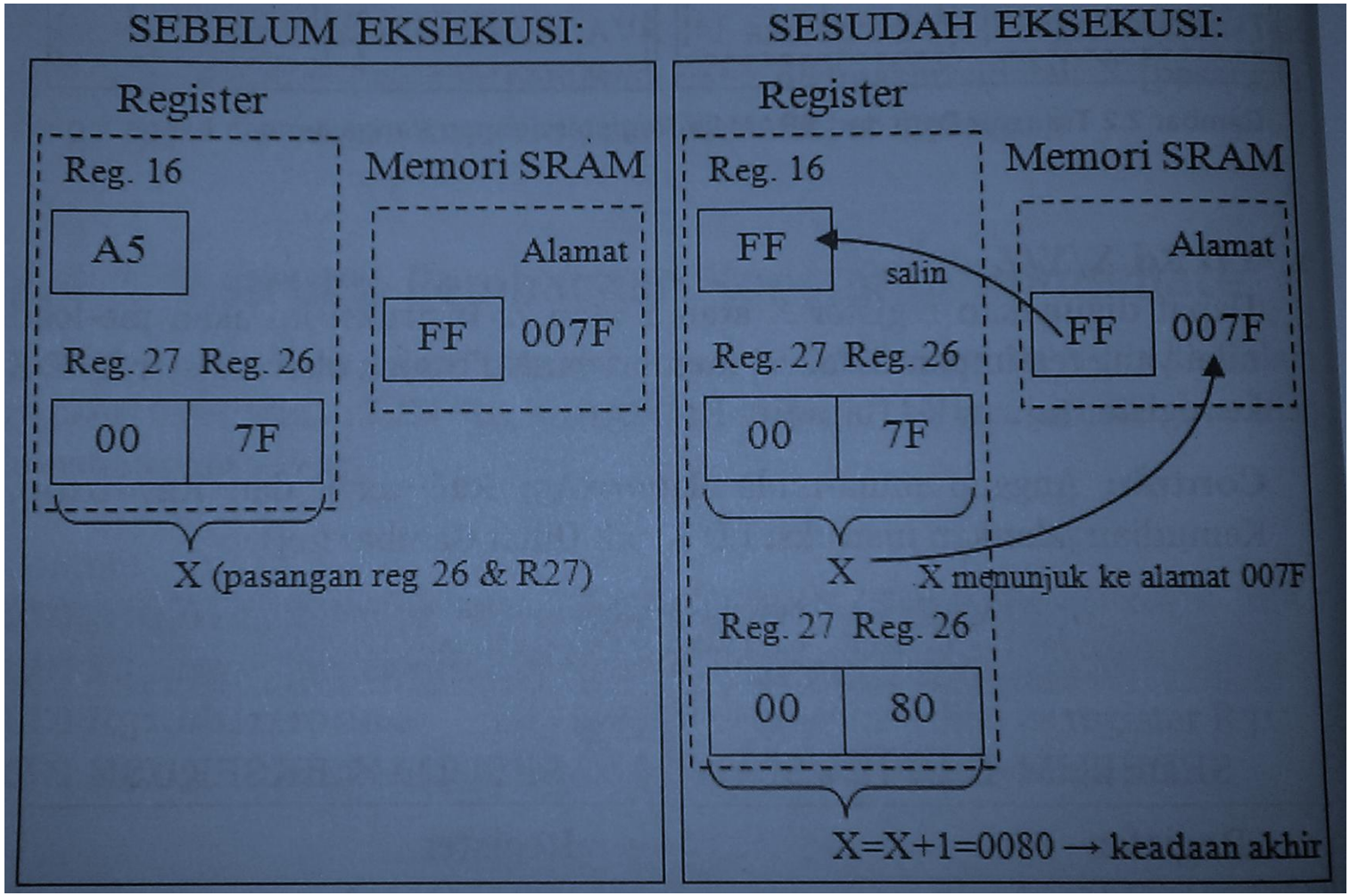
Misal:

R16=0xa5; r26=0x7F ; r27=0x00

Instruksi **LD r16,X+**

ILUSTRASI

LD r16,X+



Instruksi Pembacaan Memori SRAM

LD Rn,-X/-Y/-Z

Artinya load Rn dengan nilai yang tersimpan pada lokasi yang ditunjuk **pre-decrement** dari alamat yang tersimpan dalam X/Y/Z.

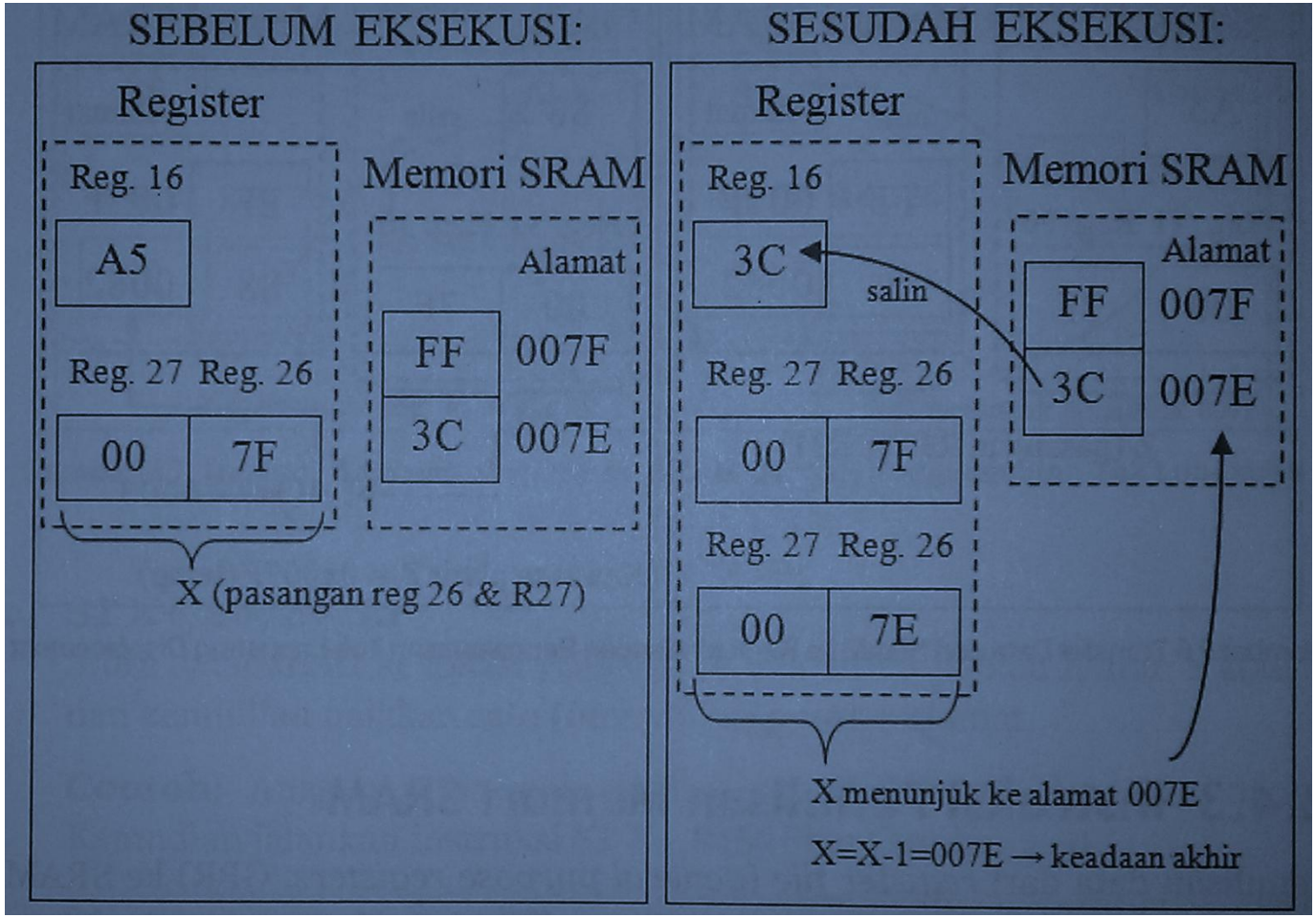
Misal:

R16=0xa5; r26=0x7F ; r27=0x00

Instruksi **LD r16,-X**

ILUSTRASI

LD r16,-X



Instruksi Pembacaan Memori SRAM

LDD Rn,Y/Z+displacement

Load Rn dengan nilai pada alamat Y atau Z+displacement. Setelah itu nilai Y atau Z tetap.

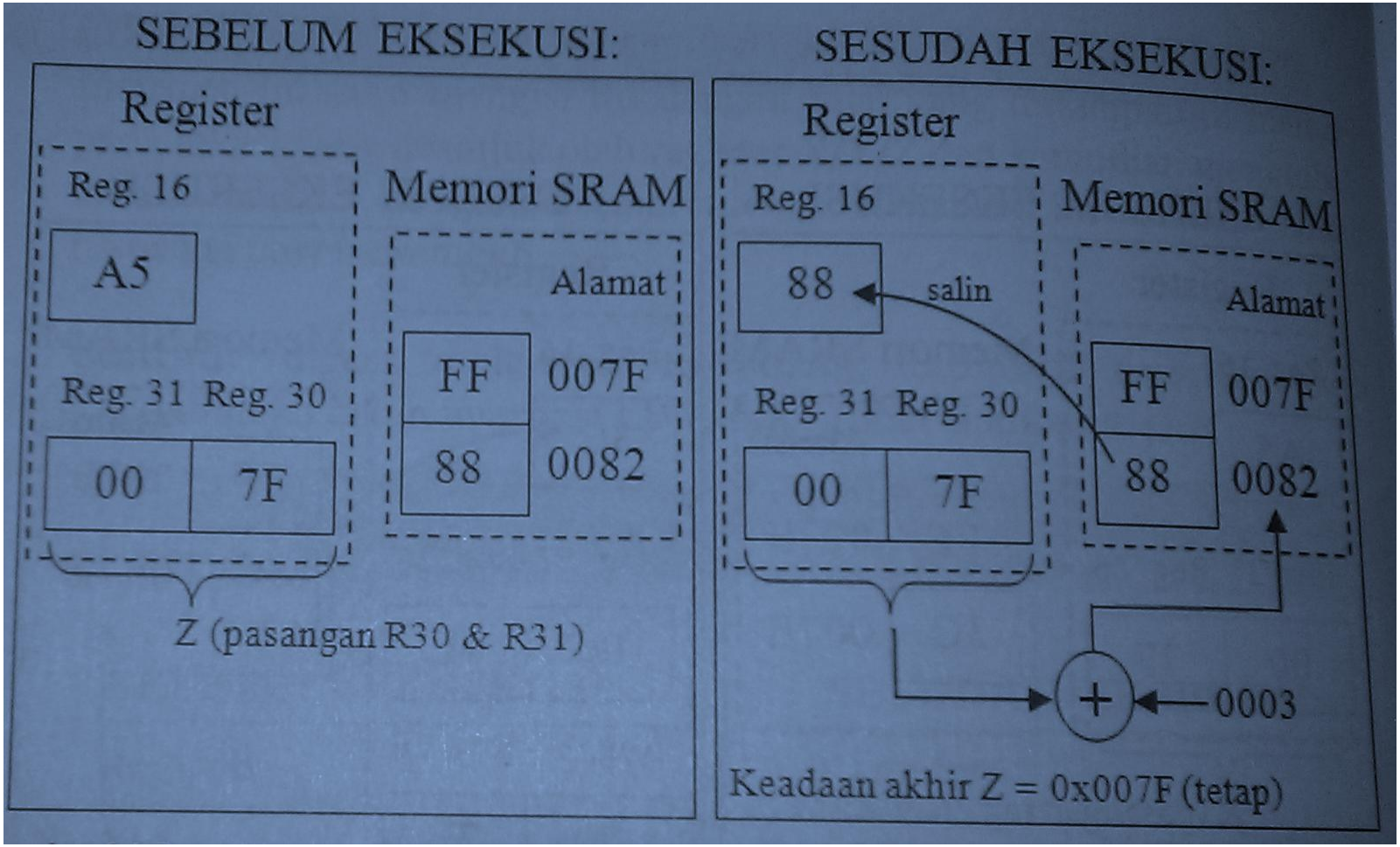
Misal:

R16=0xa5; r30=0x7F ; r31=0x00

Instruksi **LD r16,Z+0x03**

ILUSTRASI

LD r16,Z+0x03



Instruksi Penulisan Memori SRAM

Instruksi Penulisan Memori SRAM

ST X/Y/Z,Rn

Store data dari register Rn ke lokasi SRAM yang ditunjuk oleh X/Z/Y

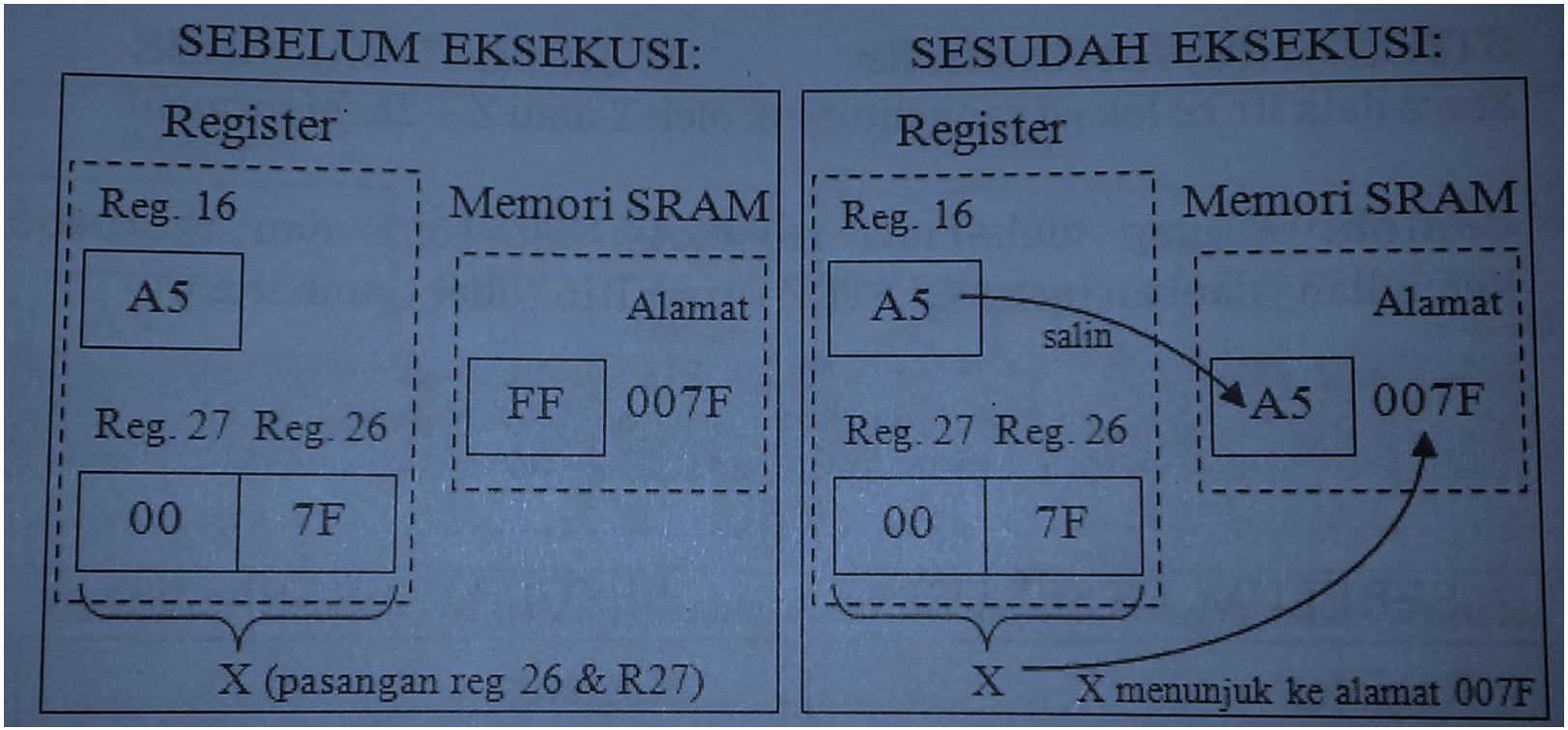
Misal:

R16=0xa5; r26=0x7F ; r27=0x00

Instruksi **ST X,r16**

ILUSTRASI

ST X,r16



Instruksi Penulisan Memori SRAM

ST X+/Y+/Z+,Rn

Store data dari Rn ke lokasi yang alamatnya ditunjuk X/Z/Y dan kemudian increment-kan pointer alamat.

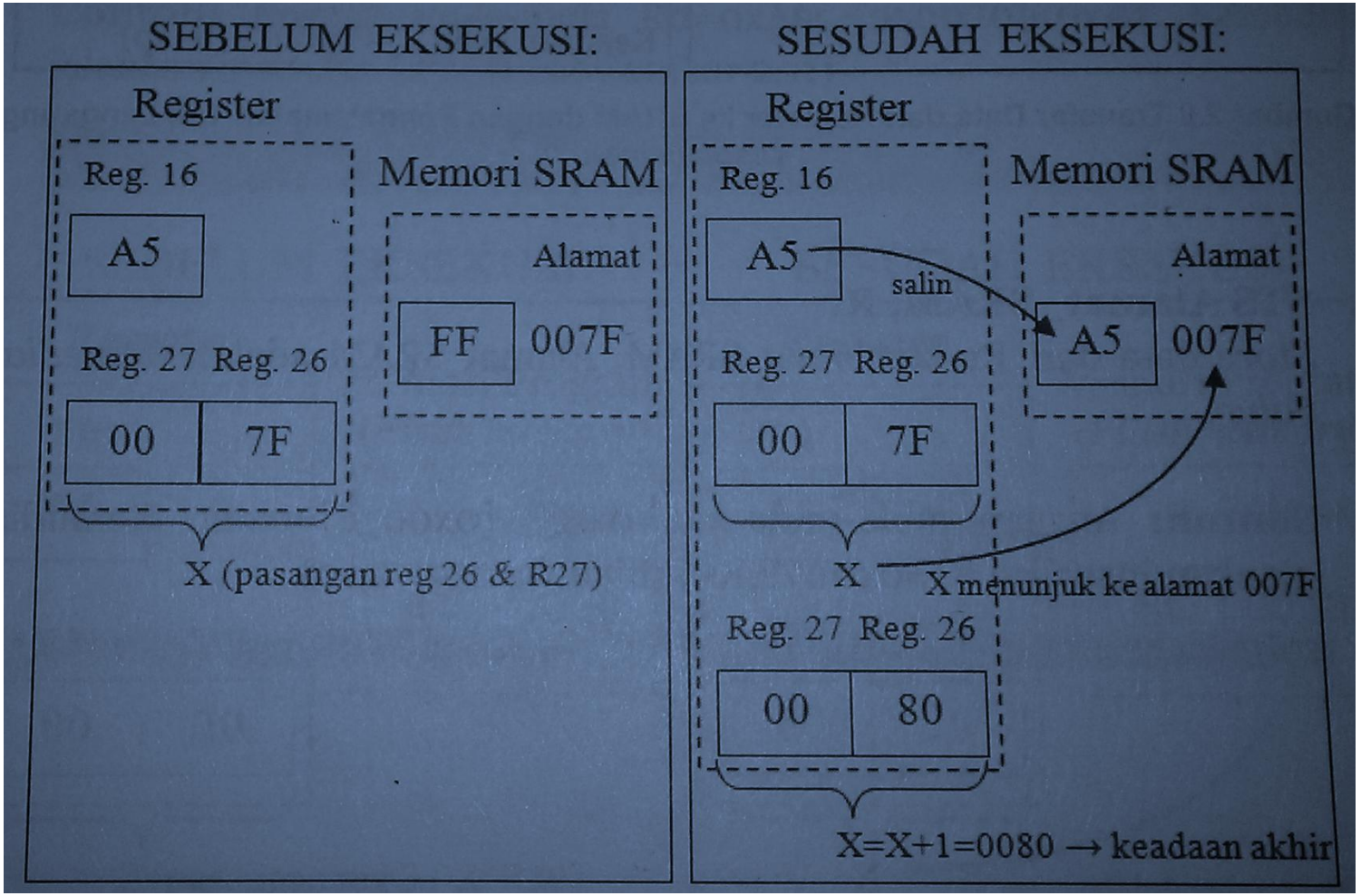
Misal:

R16=0xa5; r26=0x7F ; r27=0x00

Instruksi **ST X+,r16**

ILUSTRASI

ST X+,r16



Instruksi Penulisan Memori SRAM

STD Y/Z+displacement, Rn

Store data Rn ke lokasi yang ditunjuk oleh Y/Z+displacement. Setelah itu Y/Z nilainya tetap.

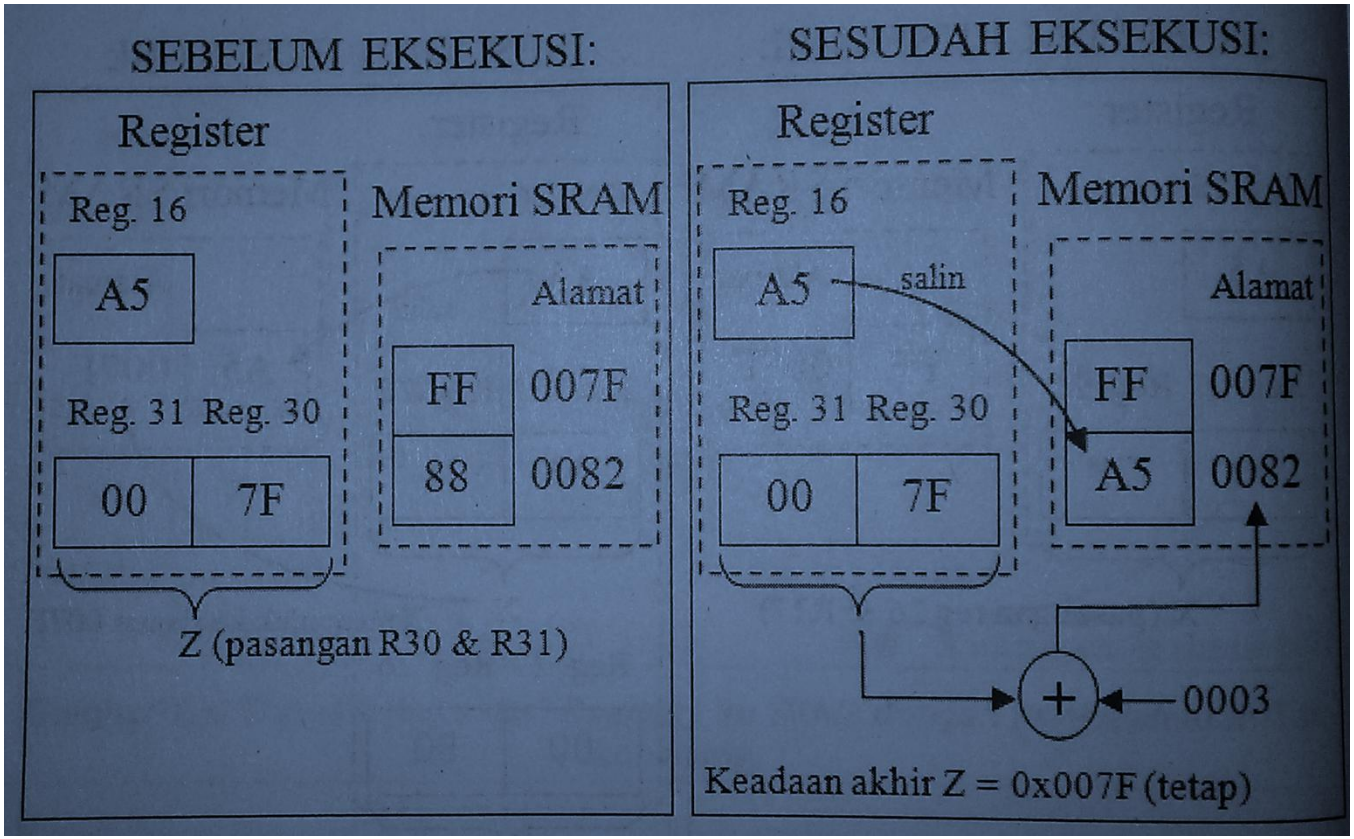
Misal:

R16=0xa5; r30=0x7F ; r31=0x00

Instruksi **STD Z+0x03,r16**

ILUSTRASI

STD Z+0x03,r16



Instruksi Penulisan Memori SRAM

STS Alamat_SRAM,Rn

Store data dari Rn ke alamat SRAM. Alamat SRAM adalah immediate value.

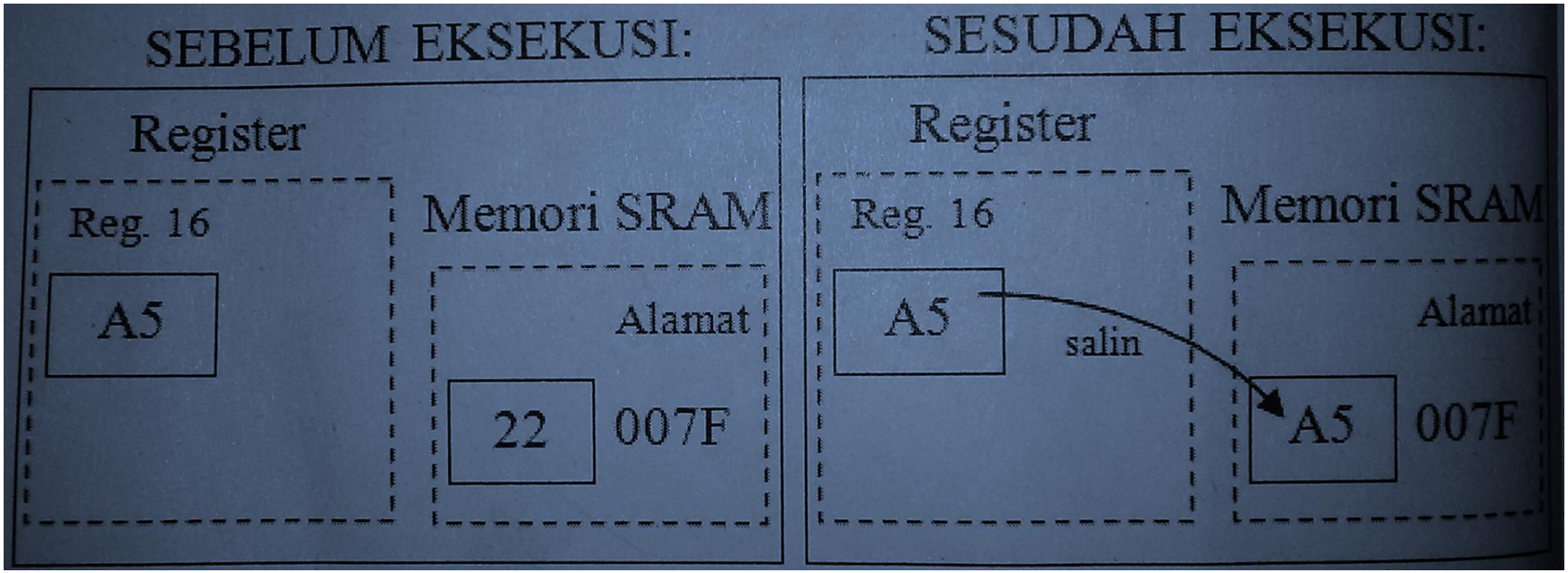
Misal:

R16=0xa5; [0x007F] = 0x22

Instruksi **STS 0x007F,r16**

ILUSTRASI

STS 0x007F,r16



Instruksi Pembacaan Memori Program

Instruksi Pembacaan Memori Program

LPM

Load data dari memori program. Instruksi ini dapat digunakan hampir semua AVR. Instruksi ini akan me-load data dari memori program (flash memori) yang alamatnya **Z ke dalam R0.**

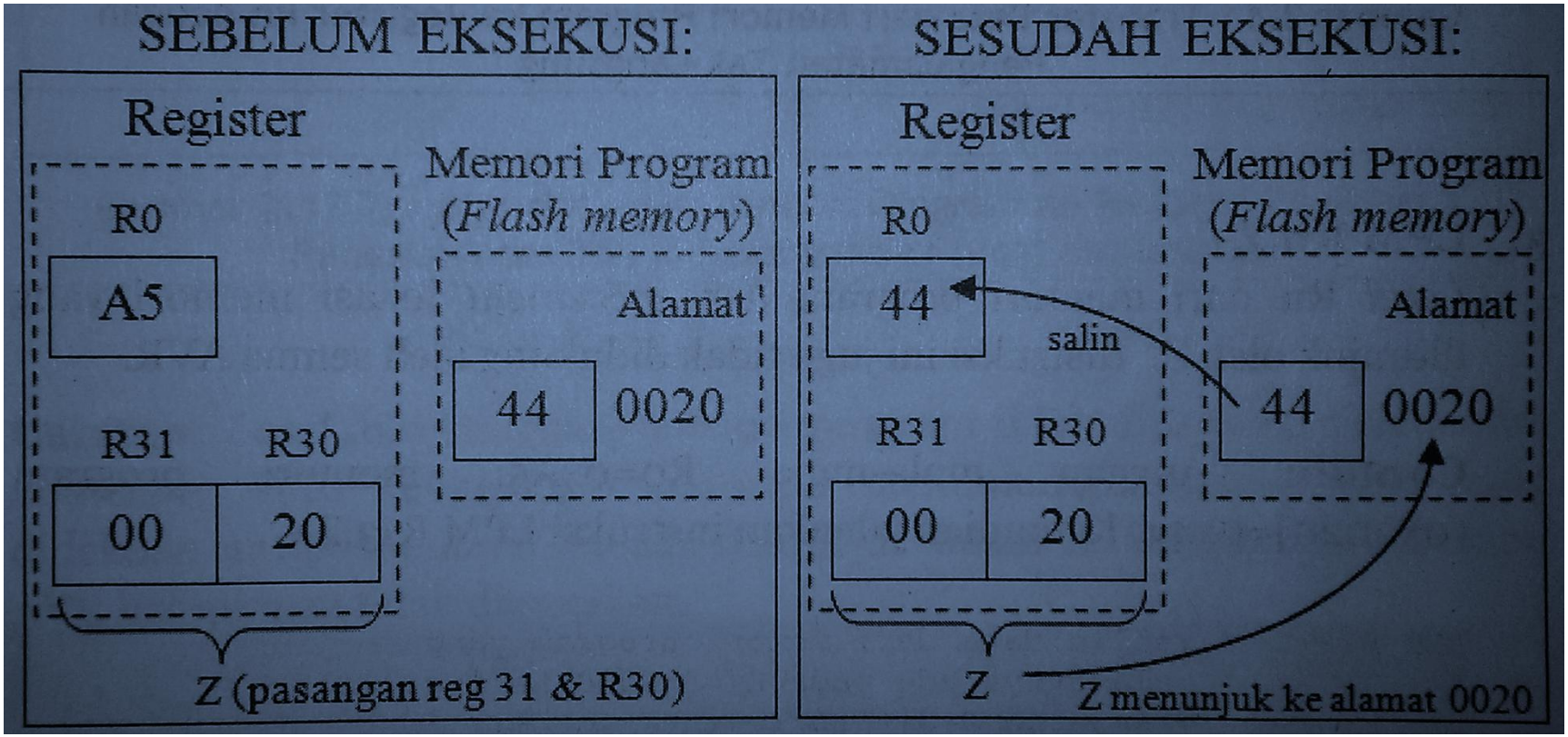
Misal:

R0=0xa5 , pointer Z [0x0020] = 0x44

Instruksi LPM

ILUSTRASI

LPM



Instruksi Pembacaan Memori Program

LPM Rn,Z

Instruksi ini mirip dengan instruksi LPM, bedanya instruksi ini dapat me-load ke semua register GPR (R0…R31).

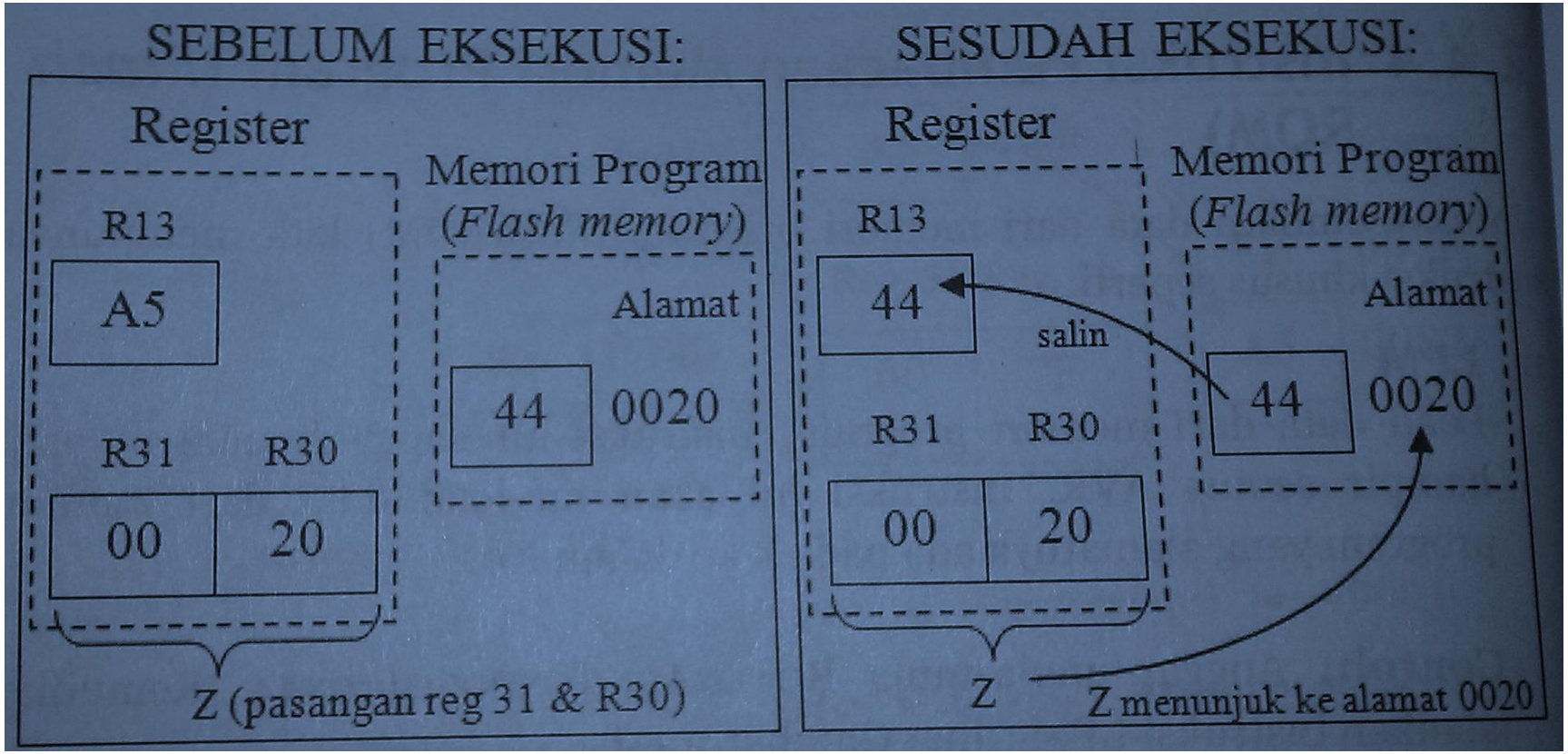
Misal:

R13=0xa5 , memori program [0x0020] = 0x44

Instruksi **LPM R13,Z**

ILUSTRASI

LPM R13,Z



Instruksi Pembacaan Memori Program

LPM Rn,Z+

Load Rn dari memori pogram dan increment lokasi memori yang ditunjuk oleh Z.

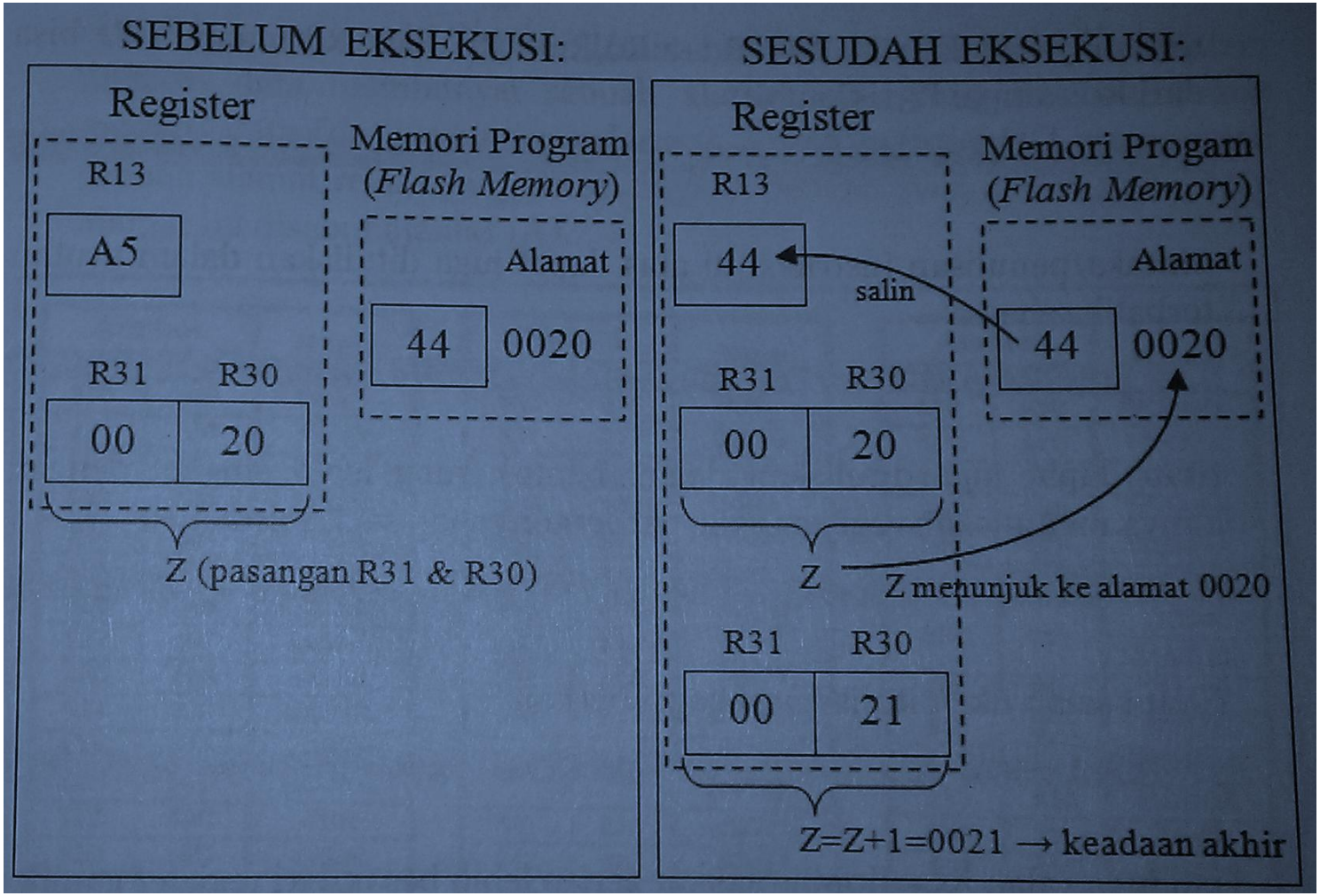
Misal:

R13=0xa5 , memori program [0x0020] = 0x44

Instruksi **LPM R13,Z+**

ILUSTRASI

LPM R13,Z+



OPERASI PORT I/O

- Operasi I/O port pada atmega8535 dapat difungsikan sebagai input atau output dengan keluaran high ataupun low.
- Untuk mengatur fungsi port I/O dapat digunakan setting DDRx dan port.
- Port I/O sebagai output hanya memberikan source current sebesar 20mA.

PORT A

- I/O Port 8 bit dua arah (bidirectional).
- Setiap pin dapat menyediakan pull up resistor.
- Output Port A memberi arus 20mA
- 8 pin Port A juga dapat digunakan untuk masukan sinyal analog.

PORT B

- I/O Port 8 bit dua arah (bidirectional).
- Setiap pin dapat menyediakan pull up resistor.
- Output Port A memberi arus 20mA

Port Pin	Fungsi Khusus
PB0	T0 = timer/counter 0 external counter input
PB1	T1 = timer/counter 1 external counter input
PB2	AIN0 = analog comparator positive input
PB3	AIN1 = analog comparator negative input
PB4	SS = SPI slave select input
PB5	MOSI = SPI bus master output/ slave input
PB6	MISO = SPI bus master input/ slave output
PB7	SCK = SPI bus serial clock

PORT C

- I/O Port 8 bit dua arah (bidirectional).
- Setiap pin dapat menyediakan pull up resistor.
- Output Port A memberi arus 20mA

- Fungsi khusus PC6 dan PC7 sebagai oscillator untuk timer/counter 2

PORT D

- I/O Port 8 bit dua arah (bidirectional).
- Setiap pin dapat menyediakan pull up resistor.
- Output Port A memberi arus 20mA

Port Pin	Fungsi Khusus
PD0	RXD (USART Input Pin)
PD1	TXD (USART Output Pin)
PD2	INT0 (External Interrupt 0 Input)
PD3	INT1 (External Interrupt 1 Input)
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD6	ICP1 (Timer/Counter1 Input Capture Pin)
PD7	OC2 (Timer/Counter2 Output Compare Match Output)

PORT I/O

- Menetapkan port sebagai input atau output tergantung pada data direction register yang dinamakan **DDRx.**
- **DDRxn = 1 ,maka Pxn digunakan sebagai pin output.**
- **DDRxn = 0 ,maka Pxn digunakan sebagai pin input.**
- Penulisan ke port register **PORTx**
- Pembacaan dari port register **PINx**

DDRxn	Portxn	I/O	Pull up	Keterangan
0	0	Input	Tidak	Tri state (Hi-Z)
0	1	Input	Ya	Pull up aktif
1	0	Output	Tidak	Output Low
1	1	Output	Tidak	Output High

CONTOH

```
;penulisan ke port
Ldi r16,$ff
Out ddra,r16        ;PORTA sebagai output
Ldi r16,$00
Out Porta,r16       ;clear semua pin-pin Port A

;pembacaan port
Ldi r16,$00
Out ddra,r16        ;PORTA sebagai input
In r16,pina         ;r16 menerima data dari PORTA
```

INSTRUKSI I/O AVR ATMEGA8535

- Operasi port I/O dapat diubah-ubah dalam program secara byte atau hanya pada bit tertentu.
- Pengubahan secara byte dilakukan dengan instruksi **in** dan **out** yang menggunakan register pembantu**.**
- Pengubahan secara bit dapat dilakukan dengan instruksi **cbi** atau **sbi.**

INSTRUKSI I/O AVR ATMEGA8535

- **in**: menulis data dari I/O space (port,timer, configuration register, dsb) kedalam register.

  Contoh: **in r16,PinA**

- **out:** menulis data dari register ke I/O space (port,timer, configuration register, dsb).

  Contoh:

  Clr r16
  Set r17
  out PortB, r16 ; menulis 0 ke portB
  Out $18,r17 ;menulis ke portB (alamat $18)

INSTRUKSI I/O AVR ATMEGA8535

- **ldi: load immediate,** menulis konstanta 8 bit langsung ke register r16 s/d r31.

  Contoh: ldi r16,0xff

- **Sbi: set bit in I/O register,** membuat logika high pada sebuah bit I/O register

  Contoh: sbi PORTB,7

INSTRUKSI I/O AVR ATMEGA8535

- **cbi: clear bit in I/O register,** membuat logika lowpada sebuah bit I/O register

Contoh: cbi PORTB,7

- **Sbic (skip if bit in I/O register is cleared):** lompati satu instruksi jika bit tunggal dalam I/O register dalam kondisi clear/low.

Contoh:

```
    Cbi portA,3
    Ldi r16,0xAA
    Sbic portA,3 ; ke skip jika portA3 low
    Ldi r16,0xBB
skip: ldi r17,0xCC
```

INSTRUKSI I/O AVR ATMEGA8535

- **Sbis (skip if bit in I/O register is set):** lompati satu instruksi jika bit tunggal dalam I/O register dalam kondisi high.

Contoh:

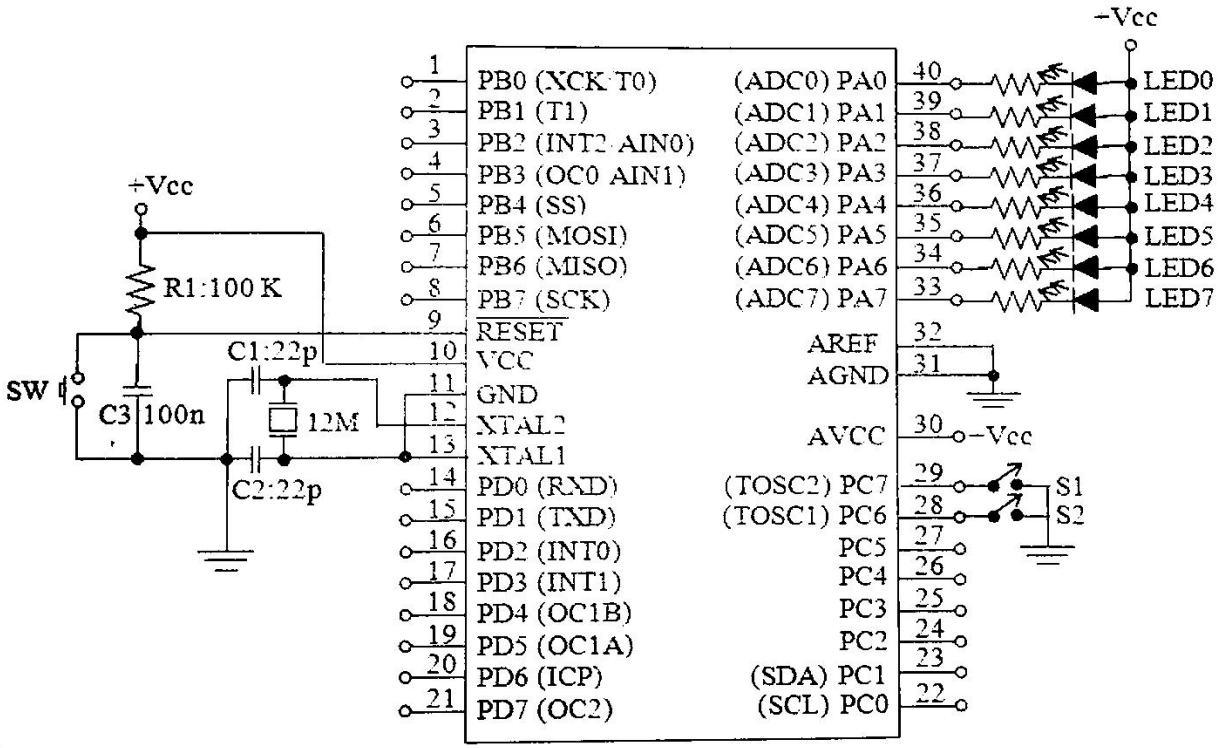
```
        sbi portA,3
        Ldi r16,0xAA
        Sbis portA,3 ; ke skip jika portA3 low
        Ldi r16,0xBB
skip:   Ldi r17,0xCC
```

LATIHAN

1. Buat program untuk mengisi register r18 dengan sebuah bilangan heksadesimal 0x66 kemudian salin nilai tersebut ke memori data alamat 0x007F s/d 0x0086 menggunakan pengalamatan tidak langsung.
2. Buat program untuk mengisi memori data alamat 0x0080 dengan sebuah bilangan decimal 250, kemudian salin kembali nilai tersebut ke register r19.
3. Buat program (dengan looping) untuk mempertukarkan data 32byte dari lokasi SRAM mulai 0x0070 s/d 0x008F (array) ke lokasi SRAM mulai 0x0090 s/d lokasi 0x00AF.
4. Buat program (dengan looping) untuk mempertukarkan data 32byte dari lokasi SRAM mulai 0x0070 s/d 0x008F (array) ke lokasi SRAM mundur mulai 0x00AF s/d lokasi 0x0090.

LATIHAN

5. Buat program apabila saklar S1 tertutup maka led ganjil menyala, tetapi jika saklar S2 tertutup maka led genap menyala. Selain itu seluruh LED menyala. (harus mengandung kode operasi **sbi,cbi,sbis dan out**).



- Buat program untuk mengisi register r18 dengan sebuah bilangan heksadesimal 0x66 kemudian salin nilai tersebut ke memori data alamat 0x007F s/d 0x0086 menggunakan pengalamatan tidak langsung.

- ldi r16,0x66
- ldi r26,0x7F
- ldi r27,0x00
- st x+,r18
- ……

- Buat program untuk mengisi memori data alamat 0x0080 dengan sebuah bilangan decimal 250, kemudian salin kembali nilai tersebut ke register r19.

- sts 0x0080,250
- lds r19,0x0080

- Buat program (dengan looping) untuk mempertukarkan data 32byte dari lokasi SRAM mulai 0x0070 s/d 0x008F (array) ke lokasi SRAM mulai 0x0090 s/d lokasi 0x00AF.

```
ldi r25,32
ldi r26,0x70
ldi r27,0x00
ldi r28,0x90
ldi r29,0x00

ulang:
ld r16,x
ld r17,y
st x+,r17
st y+,r16
dec r25
brne ulang
```

- Buat program (dengan looping) untuk mempertukarkan data 32byte dari lokasi SRAM mulai 0x0070 s/d 0x008F (array) ke lokasi SRAM mundur mulai 0x00AF s/d lokasi 0x0090.

```
ldi r25,32
ldi r26,0x70
ldi r27,0x00
ldi r28,0xB0
ldi r29,0x00

ulang:
ld r16,x
ld r17,-y
st x+,r16
st y,r17
brne ulang
```