SISTEM MIKROPROSESOR

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

OPERASI ARITMATIKA, LOGIKA, GESER DAN ROTASI

PENDAHULUAN

- Operasi Aritmatika : bilangan bertanda dan tidak bertanda.
- Operasi Logika : AND, OR, NOT dan XOR.
- Operasi geser dan rotasi : geser kanan, geser kiri, rotasi kanan dan rotasi kiri

OPERASI ARITMATIKA AVR

- Bilangan tidak bertanda data yang semua bit-ya merepresentasikan suatu nilai dann tidak ada bit yang menyatakan tanda positif ataupun negatif.
- Operand berada antara $00 s/d $FF (0 s/d 255 decimal) untuk data 8 bit.
- Operasi ALU selain memberikan hasil operasi aritmatika atau operasi logika akan memberikan hasil status kondisi register status.
- Pada umumnya setiap saat prosesor mengeksekusi sebuah instruksi akan memperngaruhi sejumlah status flag untuk merefleksikan hasil operasi yang terjadi.

PENJUMLAHAN BILANGAN TAK BERTANDA

- Operasi penjumlahan mempunyai dua buah register GPR sebagai input dan hasil operasi tersimpan pada register pertama (kiri, sebagai register tujuan).
- **Semua operasi aritmatika pada mikrokontroler AVR hanya dapat diproses oleh ALU melalui register GPR**. (tidak dapat melibatkan memori data atau penyimpanan lainnya.)

ADD (ADDITION)

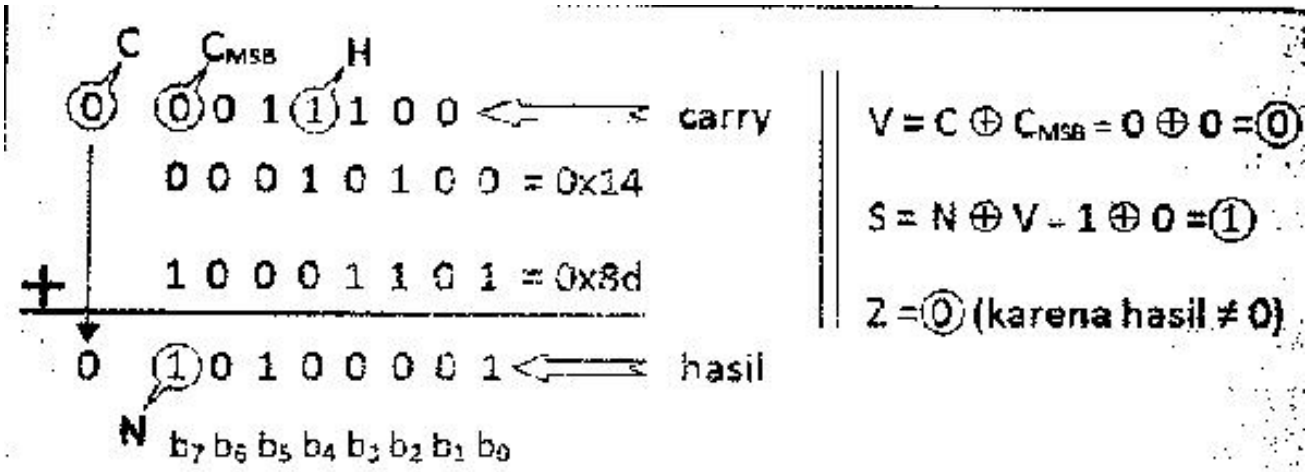
- Melakukan operasi penjumlahan antara dua register.
- Bentuk umum instruksi ADD pada AVR adalah:

ADD Rd,Rr ; Rd=Rd+Rr

- Selain hasil operasi aritmatika yang diberikan juga akan dibangkitkan flag yang terdapat pada status register (flag H,S,V,N,Z,C).

Contoh-1:

```
ldi r16,0x14
ldi r17,0x8d
add r16,r17    ;r16 = r16 + r17 = 0xa1
```



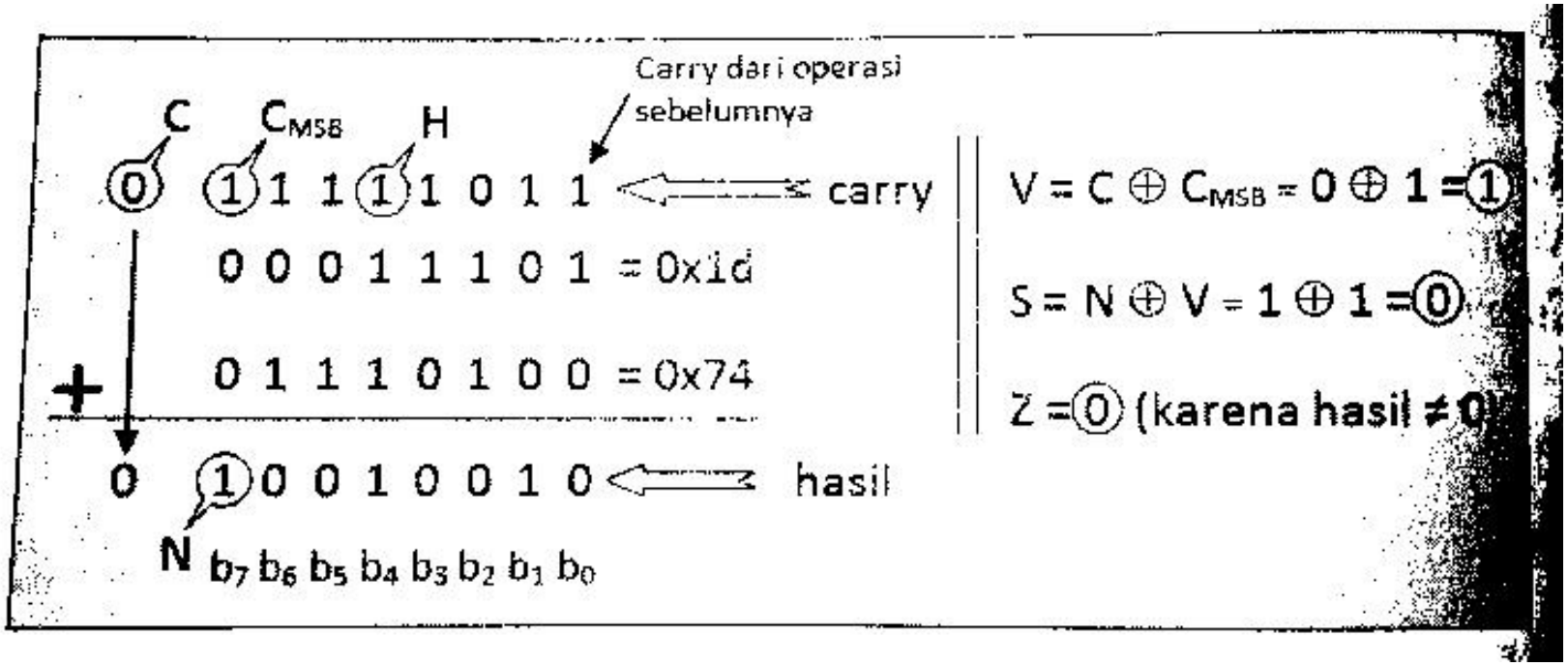
ADC (ADD WITH CARRY)

- Melakukan operasi penjumlahan antara dua register kemudian ditambah dengan carry flag yang telah ada sebelumnya.
- Bentuk umum instruksi ADC pada AVR:

ADC Rd,Rr ; Rd=Rd+Rr+C

- Register yang dapat digunakan: R0 … R31
- Selain hasil operasi aritmatika yang diberikan juga akan dibangkitkan flag yang terdapat pada status register (flag H,S,V,N,Z,C).

```
Contoh:   ldi r16,0xa9
          ldi r17,0x74
          add r16,r17     ;r16 = r16 + r17 = 0x1d; Carry = 1
          adc r16,r17     ;r16 = r16+r17+Carry = 0x1d+0x74+1 = 0x92
```



SUB (SUBRTACTION)

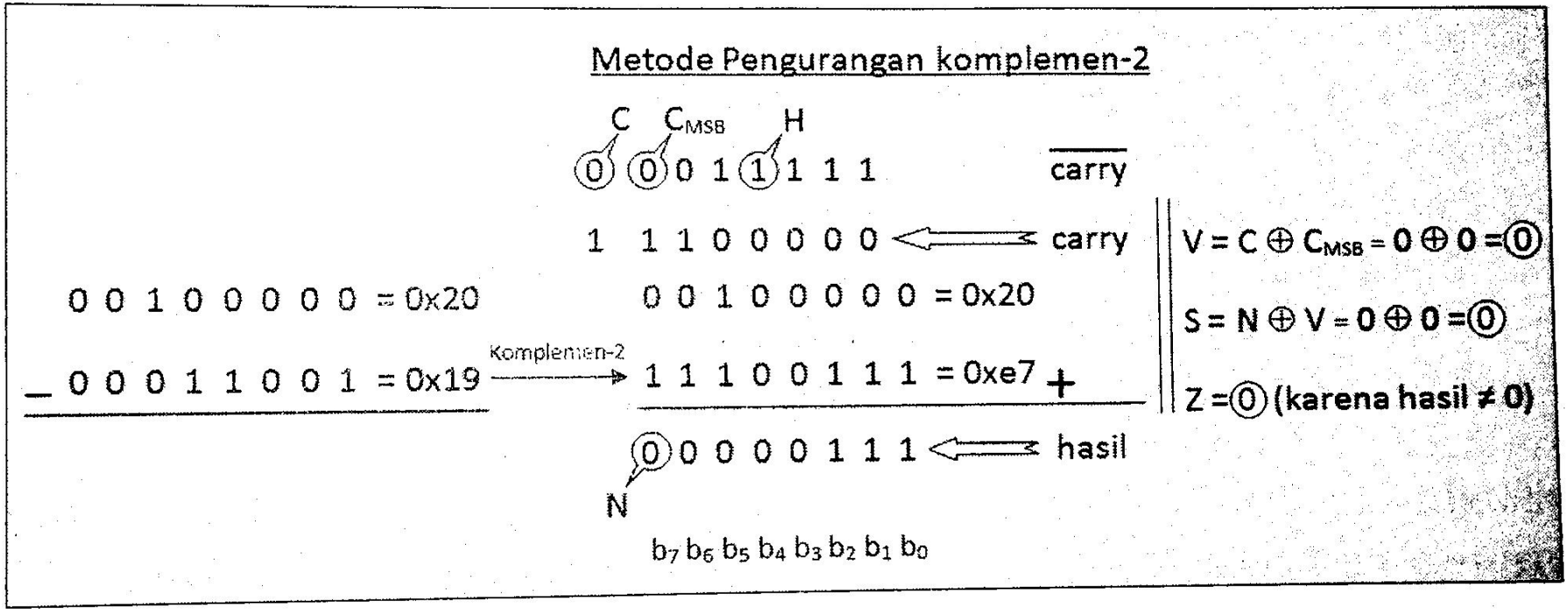
- Melakukan operasi pengurangan antara dua register.
- Bentuk umum:

SUB Rd,Rr ; Rd=Rd-Rr

- Register yang dapat digunakan: R0 … R31
- Selain hasil operasi aritmatika yang diberikan juga akan dibangkitkan flag yang terdapat pada status register (flag H,S,V,N,Z,C).

Contoh-1:

```
ldi r16,0x20
ldi r17,0x19
sub r16,r17     ;r16 = r16 - r17
```



SBC (SUBTRACT WITH CARRY)

- Melakukan operasi pengurangan antara dua register dengan carry flag yang telah ada sebelumnya.
- Bentuk umum instruksi SBC pada AVR:

SBC Rd,Rr ; Rd=Rd+Rr+C

- Register yang dapat digunakan: R0 … R31
- Selain hasil operasi aritmatika yang diberikan juga akan dibangkitkan flag yang terdapat pada status register (flag H,S,V,N,Z,C).

Contoh:

```
ldi r16,0x09
ldi r17,0x0a
sub r16,r17     ;r16 = r16 - r17 = 0xff; Carry = 1
sbc r16,r17     ;r16 = r16-r17-carry = 0xff-0x0a-1 = 0xf4
```

MUL (MULTIPLY UNSIGNED)

- Melakukan operasi perkalian dua register **8bit x 8bit** bilangan tidak bertanda.
- Hasil perkalian berupa data 16 bit yang secara otomatis tersimpan pada pasangan register r1:r0 (r0 =low byte , r1=high byte).
- Untuk memindahkan bilangan 16bit ke pasangan register yang lain maka dapat digunakan perintah **MOVW** (copy register word).

MULS (MULTIPLY SIGNED)

- Operasi perkalian dua register **8 bit x 8 bit bilangan bertanda**.
- Hasil perkalian berupa data 16 bit yang secara otomatis tersimpan pada pasangan register r1:r0 (r0 =low byte , r1=high byte).
- Untuk memindahkan bilangan 16bit ke pasangan register yang lain maka dapat digunakan perintah **MOVW** (copy register word).

MULS UNSIGNED,SIGNED

```
Contoh:     ldi r16,0x56    ;(= 86 desimal)
            ldi r17,0xfc
            muls r16,r17    ; hasil perkalian = 0xfea8
            movw r4,r0      ; salin kembali hasil ke r5:r4
```

MULSU (MULTIPLY SIGNED-UNSIGNED)

- Melakukan operasi perkalian dimana register pertama menampung bilangan bertanda 8 bit dan register kedua menampung bilangan tidak bertanda 8 bit.

MULSU Rd,Rn ; Rd signed – Rn unsigned

- Hasil perkalian berupa data 16 bit yang secara otomatis tersimpan pada pasangan register r1:r0 (r0 =low byte , r1=high byte).
- Untuk memindahkan bilangan 16bit ke pasangan register yang lain maka dapat digunakan perintah **MOVW** (copy register word).

MULSU Rd,Rn ; Rd signed – Rn unsigned

```
Contoh 1:   ldi r16,0x56          ;(= 86 desimal)
            ldi r17,0xfc
            mulsu r16,r17         ;hasil perkalian = 0x54a8
            movw r4,r0            ;salin kembali hasil ke r5:r4

Contoh 2:   ldi r16,0x56          ;(= 86 desimal)
            ldi r17,0xfc
            mulsu r17,r17         ;hasil perkalian = 0Xfea8
            movw r4,r0            ;salin kembali hasil ke r5:r4
```

- INC (INCREMENT) : Melakukan operasi penambahan 1 pada register tujuan.

  INC r22 ; r22=r22+1

- DEC (DECREMENT): Melakukan operasi pengurangan 1 pada register tujuan.

  DEC r22 ; r22=r22-1

INSTRUKSI PEMBAGIAN

- Pada mikrokontroler AVR ATMega32 tidak disediakan instruksi khusus pembagian (instruksi DIV pada keluarga MCS-51).
- Maka digunakan algoritma pembagian.

PENJUMLAHAN BILANGAN BERTANDA

- **KONSEP BILANGAN BERTANDA**

MSB							LSB
b7	b6	b5	b4	b3	b2	b1	b0
tanda	**bobot**						

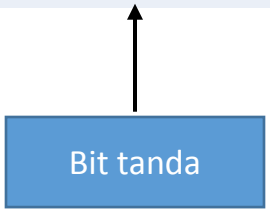
Representasi Komplemen 1

- Diperoleh dengan cara melakukan pembalikan setiap bit. Semua bit 1 diubah menjadi 0 dan semua bit 0 diubah menjadi 1.

Magnitude	Komplemen 1
0111	1000
1000	0111
1001	0110

Representasi Komplemen 1

Bilangan desimal	Representasi komplemen 1
+7	00111
-7	11000



Representasi Komplemen 1

Rentang nilai bilangan komplemen 1 :

$-(2^{n-1}-1)$ hingga $+(2^{n-1}-1)$

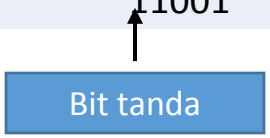
Representasi Komplemen 2

- Suatu bilangan yang diperoleh dengan menambahkan 1 pada bilangan komplemen 1.

+18	=	00010010 (twos complement)
bitwise complement	=	11101101
		+ 1
		11101110 = −18

Representasi Komplemen 2

Bilangan desimal	Representasi komplemen 2
+7	00111
-7	11001



Rentang nilai bilangan komplemen 2 :

$-(2^{n-1})$ hingga $+(2^{n-1}-1)$

Misal:
jika digunakan bilangan 8 bit maka rentang bilangannya -128 s/d +127
-128 = 1000 0000 +127=0111 1111

KONVERSI KODE BCD dan ASCII

BCD (Binary Coded Decimal):

1. Packed BCD
2. Unpacked BCD

KODE BCD

- Merupakan kode biner 4bit yang dikonversi dari bilangan decimal dasar (0 sampai 9)

Digit Desimal Dasar	Kode BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

PACKED dan UNPACKED BCD

- UNPACKED BCD

  Data disimpan 1 digit per byte. Misal 4 0000 01000

  Memerlukan memori penyimpanan 1 byte (8bit) atau register 8 bit.

- PACKED BCD

  Sebuah byte tunggal mempunyai dua digit bilangan.

  Misal 25H 0010 0101

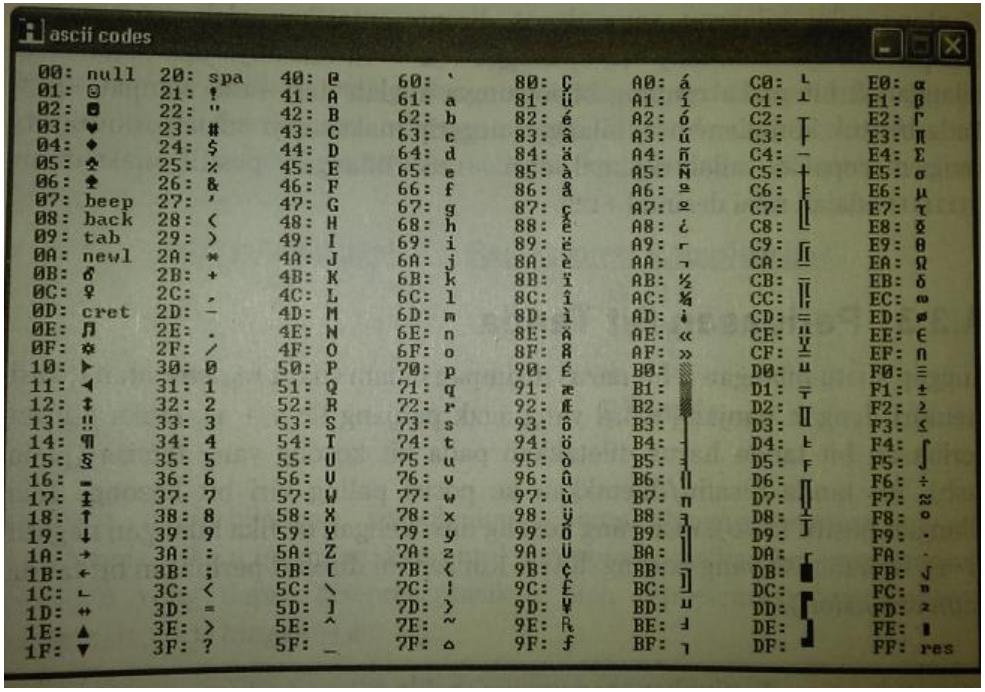
desimal	Packed BCD	Unpacked BCD
25	0010 0101	0000 0010 0000 0101
623	0000 0110 0010 0011	0000 0110 0000 0010 0000 0011
910	0000 1001 0001 0000	0000 1001 0000 0001 0000 0000

KODE ASCII

- Kode ASCII (American Standar Code For Information Interchange) mewakili karakter alphanumeric dalam memori system computer.
- Format data yang digunakan adalah 7 bit dan bit yang ke 8 digunakan untuk memuat parity dalam beberapa system.
- Jika data ASCII digunakan dalam sebuah printer maka, MSB=0 pada pencetakkan alphanumeric dan MSB=1 pada pencetakkan grafik.
- Pada PC, kumpulan extended ASCII menggunakan kode 80H-FFH.
- Karakter extended ASCII menyimpan huruf-huruf asing dan tanda baca, karakter Greek, karakter matematika, karakter box drawing dan karakter-karakter khusus lainnya.

KODE ASCII

- Pada PC, kumpulan extended ASCII menggunakan kode 80H-FFH.
- Karakter extended ASCII menyimpan huruf-huruf asing dan tanda baca, karakter Greek, karakter matematika, karakter box drawing dan karakter-karakter khusus lainnya.



KONVERSI PACKED BCD KE ASCII

- Untuk mengkonversi packed BCD ke ASCII, maka pertama kita harus mengubahnya dahulu ke unpacked BCD, lalu unpacked BCD ditambahkan dengan 0011 0000 (30H).

Packed BCD	Unpacked BCD	ASCII
45H	04H dan 05H	34H dan 35H
19H	01H dan 09H	31H dan 39H
01H	00H dan 01H	30H dan 31H

- Buat program

KONVERSI ASCII KE PACKED BCD

- Untuk mengubah ASCII ke packed BCD maka pertama yang kita lakukan adalah mengubah ASCII ke unpacked BCD (menghilangkan angka 3) dan menggabungkannya menjadi packed BCD.

Key	ASCII	Unpacked BCD	Packed BCD
5	35	05	**58H**
8	38	08	

- Buat program

OPERASI LOGIKA DAN PERGESERAN

- Nilai biner 0 = salah dan nilai biner 1 = benar.

a	b	a AND b	a OR b	a XOR b	NOT a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

CONTOH

	1010 1010
AND	1111 0000
=	1010 0000

	1010 1010
OR	1111 0000
=	1010 1010

	1010 1010
XOR	1111 0000
=	0101 1010

	1010 1010
NOT	0101 0101

INSTRUKSI AND, OR, EOR, COM DAN NEG

FORMAT:

- AND operand_tujuan, operand_sumber
- OR operand_tujuan, operand_sumber
- EOR operand_tujuan, operand_sumber
- COM operand_tujuan
- NEG operand_tujuan

Operand sumber dan operand tujuan keduanya dapat berupa R0..R31.

Hasil operasi disimpan pada **operand_tujuan.**

INSTRUKSI AND, OR, EOR, COM DAN NEG

- Salah satu penggunaan dari AND, OR dan XOR adalah melakukan perubahan bit secara selektif pada **operand tujuan**.
- Untuk melakukan hal tersebut kita membuat sebuah pola bit operand sumber yang dikenal dengan **bitmask**.
- Bitmask dipilih agar bit operand tujuan yang berhubungan diubah berdasarkan keinginan bila instruksi dieksekusi.

INSTRUKSI AND, OR, EOR, COM DAN NEG

- Untuk memilih bitmask dapat menggunakan sifat dari AND, OR dan XOR berdasakan table kebenaran masing-masing.

Ø AND 1 = Ø	Ø OR 0 = Ø	Ø XOR 0 = Ø
Ø AND 0 = 1	Ø OR 1 = 1	Ø XOR 1 = ~Ø (Komplemen Ø)

- Ø = artinya dapat berlogika 0 atau 1

INSTRUKSI AND, OR, EOR, COM DAN NEG

1. Instruksi AND dapat digunakan untuk meng-**clear** secara spesifik **bit tujuan** sedangkan yang lainnya tetap. Bitmask 0 meng-clear bit tujuan yang berhubungan dan Bitmask 1 menjaga/membiarkan bit tujuan yang berhubungan.
2. Instruksi OR dapat digunakan untuk men-**set** secara spesifik bit **tujuan** sedangkan bit lainnya tetap. Bitmask 1 meng-**set** bit tujuan yang berhubungan dan Bitmask 0 menjaga/membiarkan bit tujuan yang berhubungan.
3. Instruksi XOR dapat digunakan untuk meng-**complement-**kan secara spesifik bit tujuan sedangkan yang lainnya tetap. Bitmask 1 meng-complement bit tujuan yang berhubungan dan Bitmask 0 menjaga/membiarkan bit tujuan yang berhubungan.

INSTRUKSI AND, OR, EOR, COM DAN NEG

1. Clear bit tanda pada register r16 sedangkan bit yang lainnya tetap.

- Solusi: gunakan instruksi AND dengan bit 0111 1111 = 7FH sebagai **BITMASK.**

ANDI r16,7FH

2. Set bit MSB dan LSB pada register r16 sementara bit yang lainnya tetap.

- Solusi: gunakan instruksi OR dengan bit 1000 0001 = 81H sebagai **BITMASK.**

ORI r16,81H

INSTRUKSI AND, OR, EOR, COM DAN NEG

3. Ubah bit tanda pada register r16

- Solusi: gunakan instruksi EOR dengan byte 0x80 sebagai BITMASK.

LDI r16,$80

EOR r16,r17

INSTRUKSI AND, OR, EOR, COM DAN NEG

AND

- Melakukan operasi logika AND antara dua register.
- Contoh:

```
LDI R16,0x80
LDI R17,0x45
AND R17,R16  ;melindungi bit tanda pada R17
```

- Status flag yang terpengaruh pada operasi logika AND adalah S=Ø,

  N=Ø, Z=Ø dan V=0.

INSTRUKSI AND, OR, EOR, COM DAN NEG

ANDI

- Melakukan operasi logika AND antara register dengan immediate data.
- Contoh:

```
ANDI R16,$0F        ; clear 4 bit atas r17
ANDI R17,$10        ; isolasi bit 4 pada r18
ANDI R18,$AA        ; clear bit-bit ganjil pada r19
```

- Status flag yang terpengaruh pada operasi logika AND adalah S=Ø,

  N=Ø, Z=Ø dan V=0.

INSTRUKSI AND, OR, EOR, COM DAN NEG

OR

- Melakukan operasi logika OR antara dua register
- Contoh:

```
Ldi r16,0x80
Ldi r17,0x45
OR r17,r16  ; memaksa bit tanda dari r17 menjadi high
```

- Status flag yang terpengaruh pada operasi logika AND adalah S=Ø,

  N=Ø, Z=Ø dan V=0.

INSTRUKSI AND, OR, EOR, COM DAN NEG

ORI

- Melakukan operasi logika OR antara register dengan immediate data.
- Contoh:

```
ORI r16,0xF0        ;set high 4 bit atas r16
ORI r17,1           ;set high bit 0 pada r17
```

- Status flag yang terpengaruh pada operasi logika AND adalah S=Ø,

  N=Ø, Z=Ø dan V=0.

INSTRUKSI AND, OR, EOR, COM DAN NEG

EOR

- Melakukan operasi logika XOR antara dua register
- Contoh:

```
LDI r16,0x80
LDI r17,0x45
EOR r17,r16          ;membalik bit tanda (b7) pada r17
```

INSTRUKSI AND, OR, EOR, COM DAN NEG

COM (COMPLEMENT)

- Melakukan operasi logika komplemen 1 pada register
- Contoh:

```
LDI r16,0xF0        ;set 4 bit atas r16
COM r16             ;balik semua bit (r16=$0F)
```

INSTRUKSI AND, OR, EOR, COM DAN NEG

NEG (NEGATE)

- Melakukan operasi logika komplemen 2 pada register
- Contoh:

```
LDI r16,0xF0        ;set 4 bit atas r16
NEG r16             ;balik semua bit lalu +1 (r16=$10)
```

INSTRUKSI GESER DAN ROTASI

FORMAT:

- LSR register_tujuan
- LSL register_tujuan
- ROR register_tujuan
- ROL register_tujuan
- ASR register_tujuan

INSTRUKSI GESER DAN ROTASI

- LSR (LOGICAL SHIFT RIGHT)

Geser semua bit dalam register satu tempat ke kanan, bit 7 clear, bit 0 digeser kedalam C flag (SREG).


```
Contoh:     ldi r16,0b11000001
            lsr r16             ;  r16 = 0b01100000, C = 1
```

INSTRUKSI GESER DAN ROTASI

- LSR (LOGICAL SHIFT LEFT)

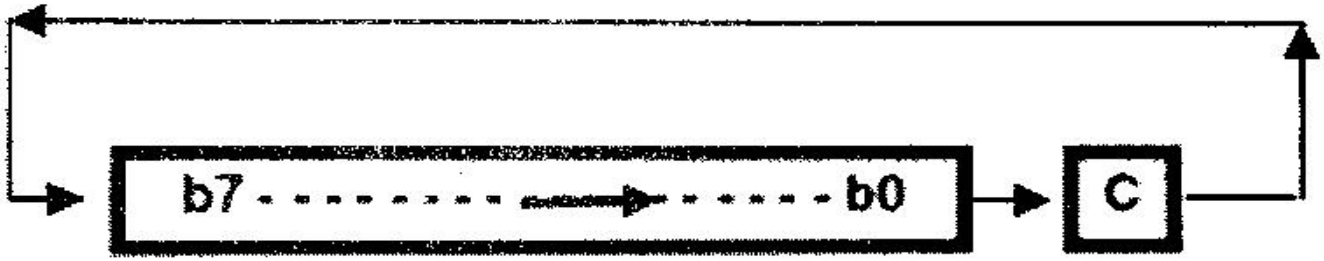
Geser semua bit dalam register satu tempat ke kiri, bit 0 clear, bit 7 digeser kedalam C flag (SREG).


```
Contoh:     ldi r16,0b11000001
            lsl r16          ;  r16 = 0b10000010, C = 1
```

INSTRUKSI GESER DAN ROTASI

- ROR (ROTATE RIGHT THROUGH CARRY)

Geser semua bit dalam register satu tempat ke kanan, C Flag digeser ke bit 7, bit 0 digeser kedalam



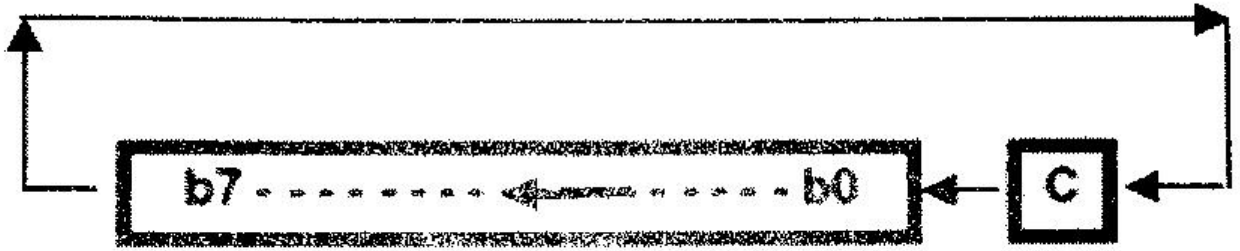
Contoh:

```
ldi r16,0b11100001
clc                     ; clear carry flag (C=0)
ror r16                 ;  r16 = 0b01110000, C = 1
```

INSTRUKSI GESER DAN ROTASI

- ROL (ROTATE LEFT THROUGH CARRY)

Geser semua bit dalam register satu tempat ke kiri, C Flag digeser ke bit 0, bit 7 digeser kedalam C flag




```
Contoh:     ldi r16,0b01100001
            sec                         ; set carry flag (C=1)
            rol r16                     ; r16 = 0b11000011, C = 0
```

LATIHAN

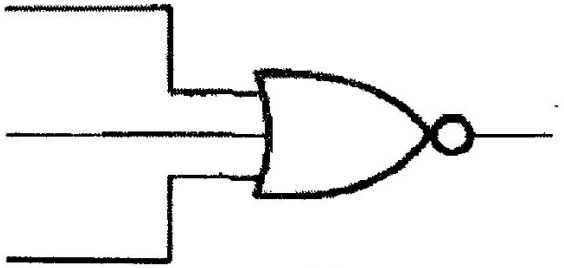
1. r5=(r2+10)*(5+r3) ; r3 bilangan bertanda
2. r16=r16-10*r3
3. Hitung dan gunakan pengalamatan immediate

0x25FF9234

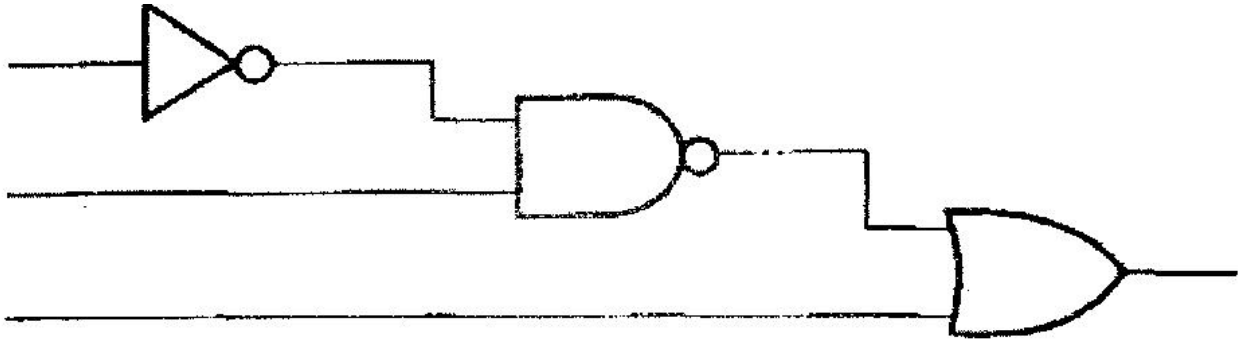
0x43976213 **+**

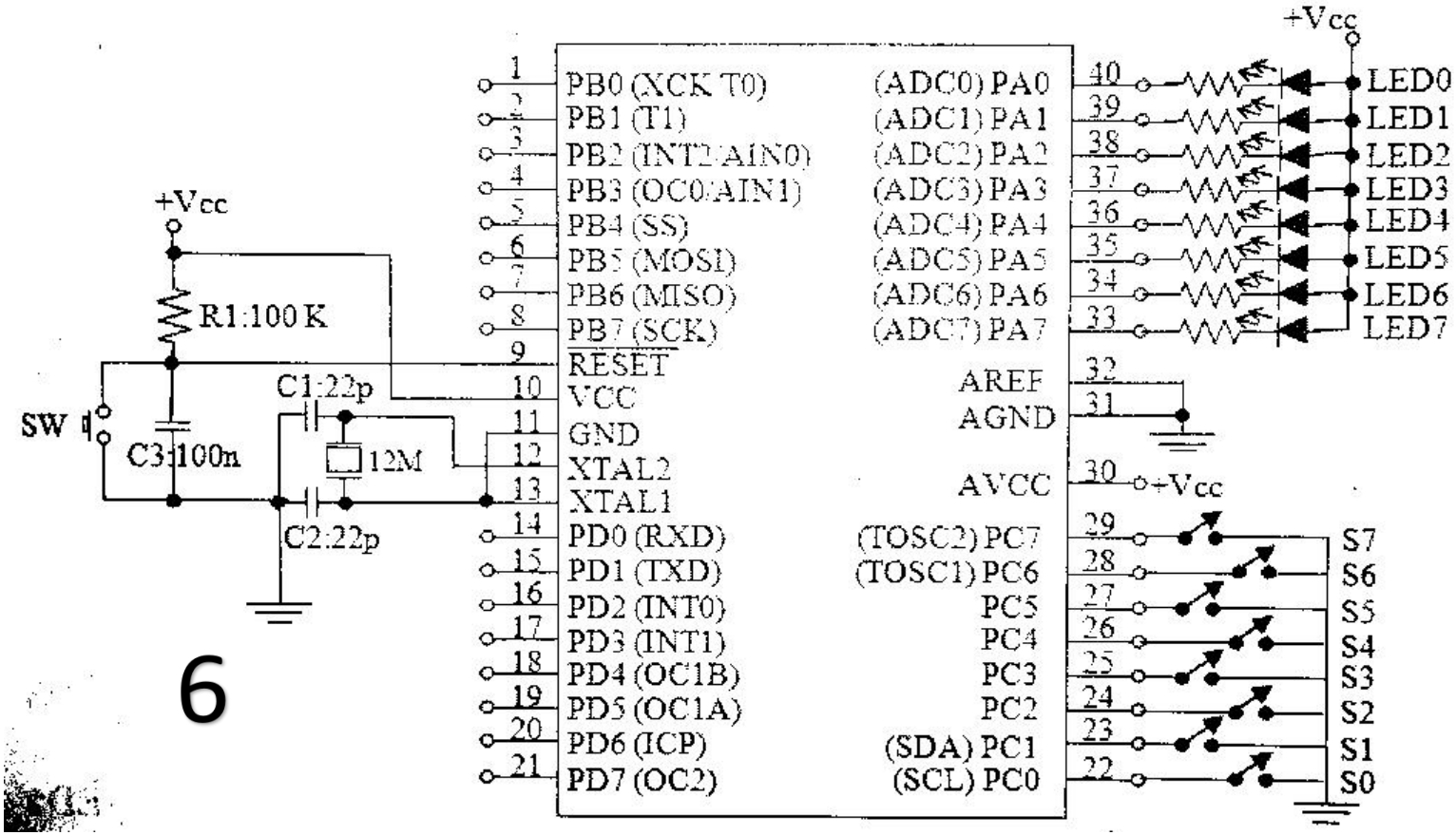
R16R17R18R19

4



5




+Vcc
R1:100 K
SW
C3:100n
C1:22p
12M
C2:22p
1 PB0 (XCK T0)
PB1 (T1)
PB2 (INT2 AIN0)
4 PB3 (OC0 AIN1)
PB4 (SS)
6 PB5 (MOSI)
PB6 (MISO)
8 PB7 (SCK)
9 RESET
10 VCC
11 GND
12 XTAL2
13 XTAL1
14 PD0 (RXD)
15 PD1 (TXD)
16 PD2 (INT0)
17 PD3 (INT1)
18 PD4 (OC1B)
19 PD5 (OC1A)
20 PD6 (ICP)
21 PD7 (OC2)
(ADC0) PA0 40
(ADC1) PA1 39
(ADC2) PA2 38
(ADC3) PA3 37
(ADC4) PA4 36
(ADC5) PA5 35
(ADC6) PA6 34
(ADC7) PA7 33
AREF 32
AGND 31
AVCC 30 +Vcc
(TOSC2) PC7 29
(TOSC1) PC6 28
PC5 27
PC4 26
PC3 25
PC2 24
(SDA) PC1 23
(SCL) PC0 22
+Vcc
LED0
LED1
LED2
LED3
LED4
LED5
LED6
LED7
S7
S6
S5
S4
S3
S2
S1
S0

6