



Pengenalan AVR

Bayu Anggoroajati, Grafika Jati, M. Anwar Ma'sum

Pembahasan Bab Ini

- Arsitektur AVR
- Register
- Perintah di AVR
- Contoh program AVR

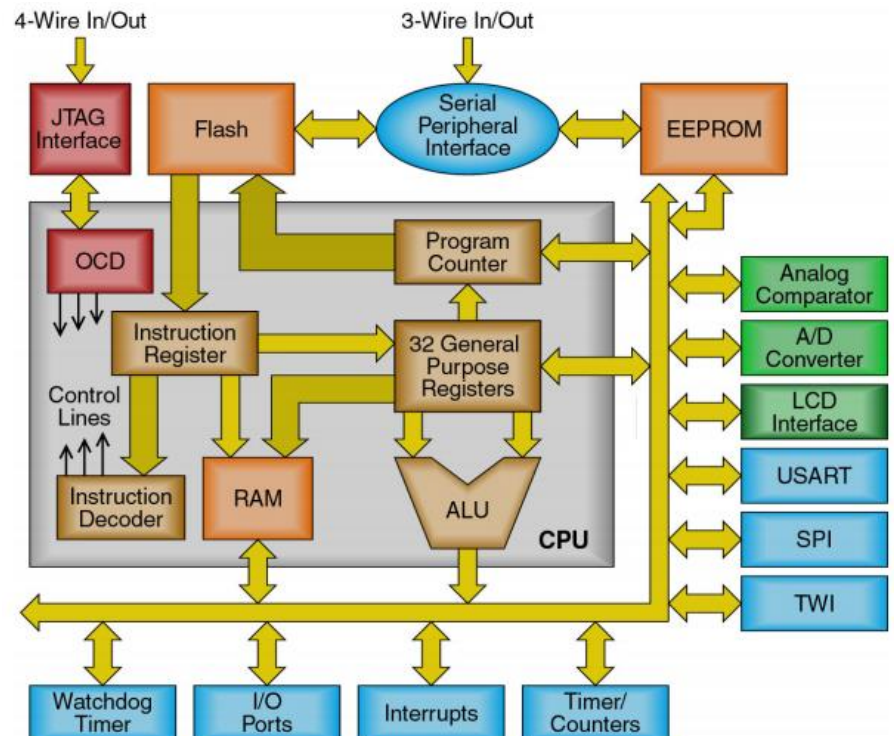
AVR

- Dikembangkan oleh **A**lf-Egil Bogen and **V**egard Wollan
- Merupakan **R**ISC processor
- **A**vanced **V**irtual **R**isc processor

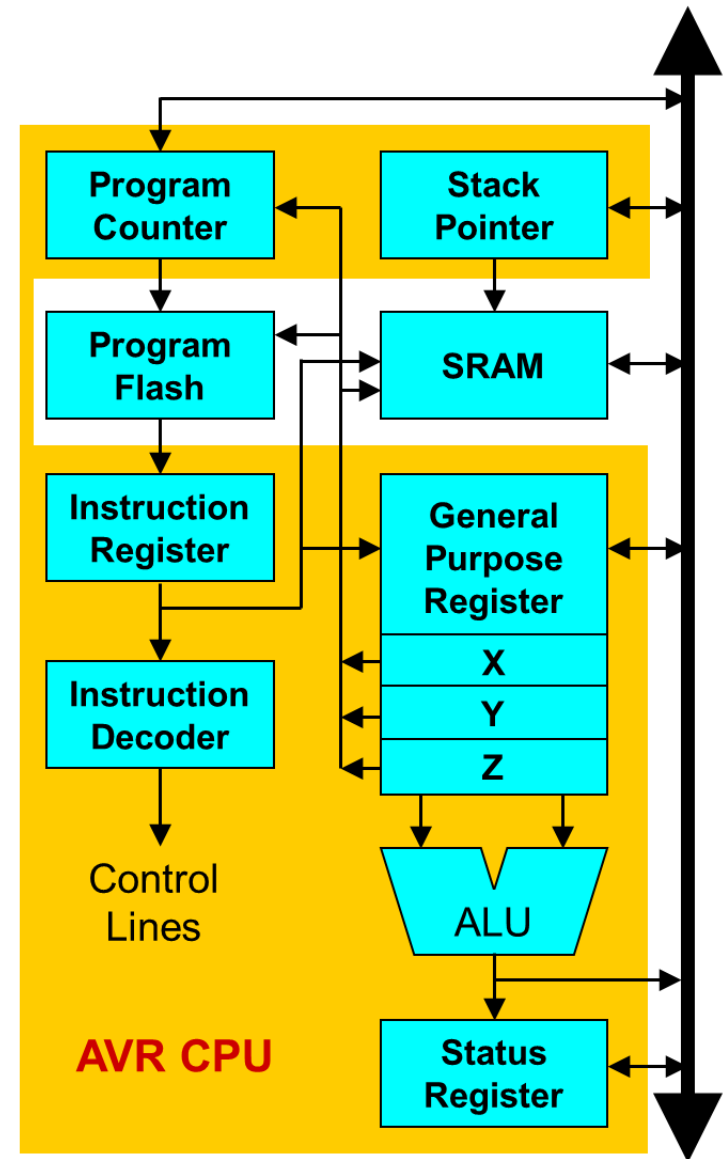


AVR

- Single cycle execution
 - One instruction per external clock
 - Low power consumption
- 32 Working Registers
 - All Directly connected to ALU!



Arsitektur AVR

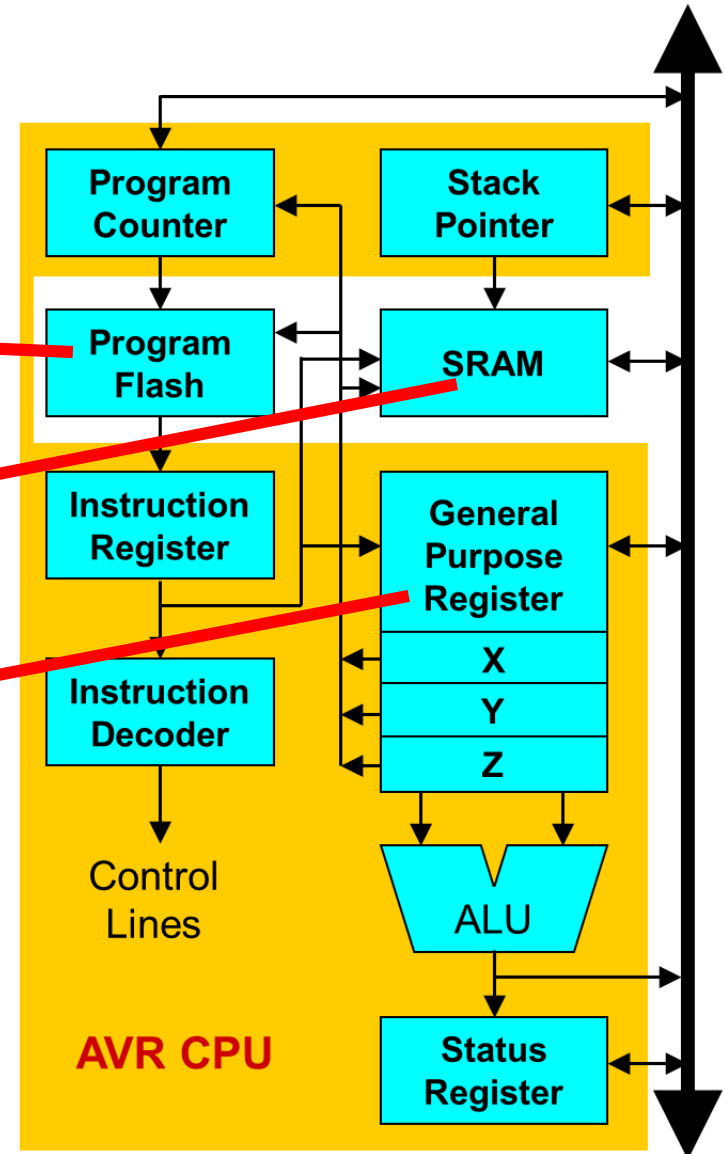


Arsitektur AVR

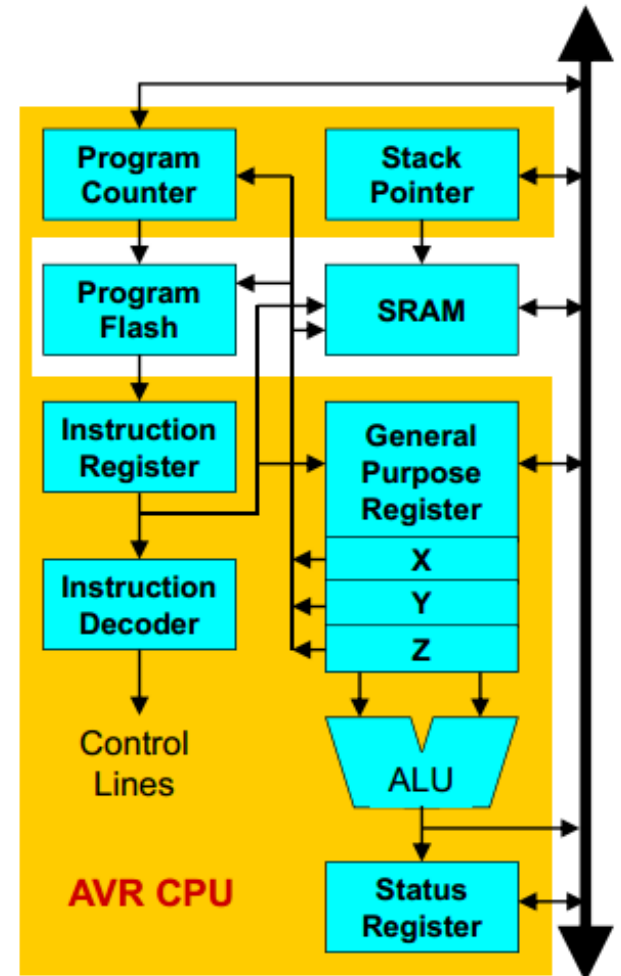
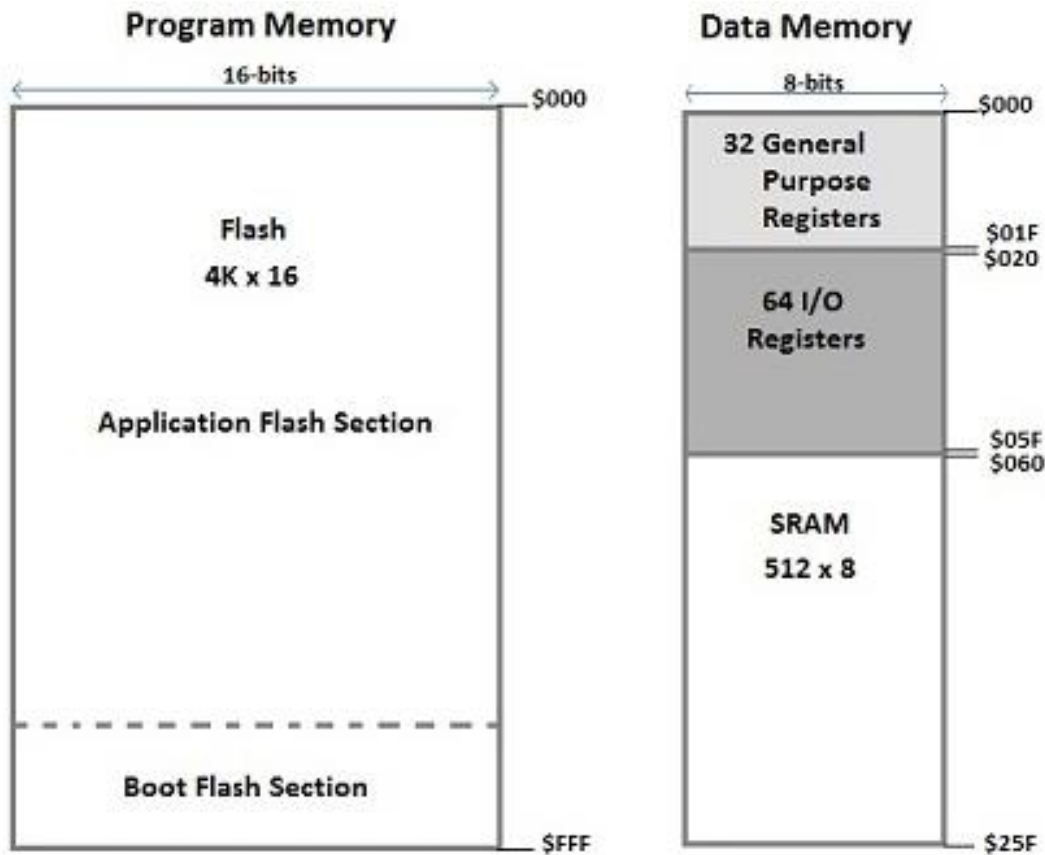
Program memory

Data memory

Register



Arsitektur AVR



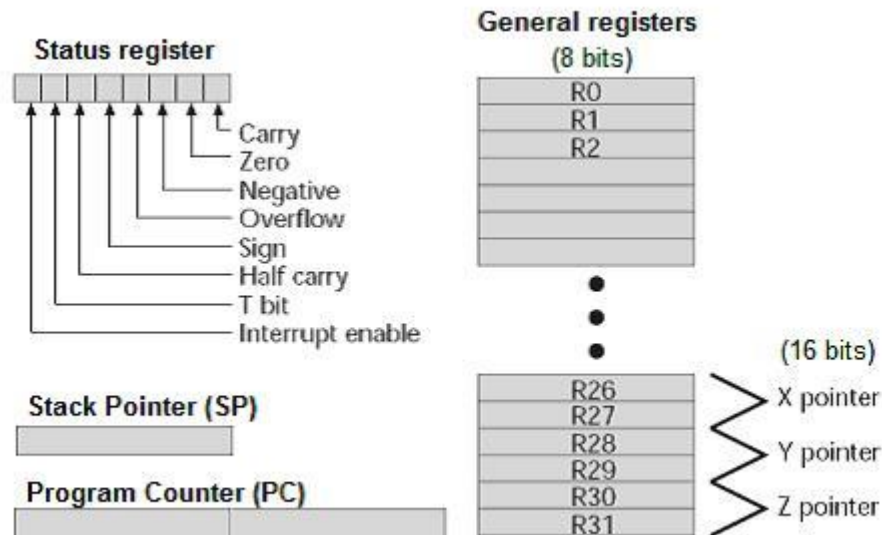
AVR Register

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

- Menggunakan 8 bit register
- Dapat menyimpan:
 - Angka dari 0 sampai 255 (unsigned),
 - Angka dari -128 to +127 (signed),
 - ASCII-coded character
 - Digit 8 bit apapun

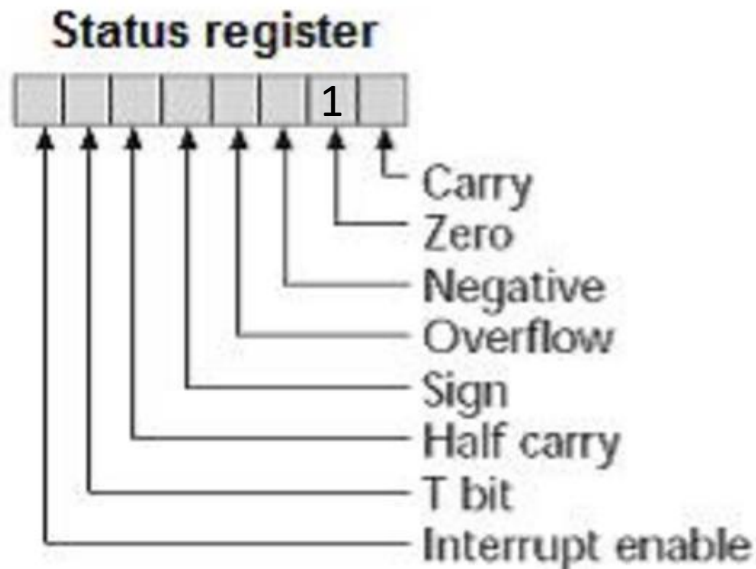
AVR Register

- Register di AVR terdapat beberapa jenis:
 - 1 Program Counter (16 bit)
 - 1 Status Register (8 bit)
 - Stack Pointer (8 bit)
 - 32 General Purpose Register (8 bit)



AVR Register: Status Register

- Berisi 8 flag (indikator) yang berbeda.
- Flag akan diperbarui berdasarkan hasil dari setiap operasi di CPU
- Contoh:
 - Jika operasi di CPU menghasilkan nilai 0, maka flag 'zero' akan ditetapkan dengan nilai 1



AVR Register: General Purpose

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

- Karakteristik register:
 - Bisa digunakan langsung pada perintah assembler
 - Operasi pada nilai register hanya membutuhkan sebuah perintah
 - Register terhubung langsung ke CPU
 - Register merupakan sumber dan tujuan dari nilai hasil perhitungan
 - 6 Register terakhir (R26-R31) dapat dikombinasikan sebagai tiga register yang memiliki panjang 16-bit (Register X, Y, dan Z)

AVR Register

0x00		R0
0x01		R1
0x02		R2
0x03		R3
0x04		R4
0x05		R5
0x06		R6

•
•
•

0x1A	XL	R26
0x1B	XH	R27
0x1C	YL	R28
0x1D	YH	R29
0x1E	ZL	R30
0x1F	ZH	R31

AVR memiliki 32 register

R0, R1, R2, ... R31

AVR Register

0x00		R0
0x01		R1
0x02		R2
0x03		R3
0x04		R4
0x05		R5
0x06		R6

•
•
•

0x1A	XL	R26
0x1B	XH	R27
0x1C	YL	R28
0x1D	YH	R29
0x1E	ZL	R30
0x1F	ZH	R31

AVR memiliki 32 register

R0, R1, R2, ... R31

Contoh 1:

```
LDI R16,150
```

AVR Register

0x00		R0
0x01		R1
0x02		R2
0x03		R3
0x04		R4
0x05		R5
0x06		R6

•
•
•

0x1A	XL	R26
0x1B	XH	R27
0x1C	YL	R28
0x1D	YH	R29
0x1E	ZL	R30
0x1F	ZH	R31

AVR memiliki 32 register

R0, R1, R2, ... R31

Contoh 1:

```
LDI R16,150
```

Contoh 2:

```
LDI R16,150
```

```
MOV R15, R16
```

AVR Register

0x00		R0
0x01		R1
0x02		R2
0x03		R3
0x04		R4
0x05		R5
0x06		R6

•
•
•

0x1A	XL	R26
0x1B	XH	R27
0x1C	YL	R28
0x1D	YH	R29
0x1E	ZL	R30
0x1F	ZH	R31

Contoh 3:

```
LDI R15,150
```

AVR Register

0x00		R0
0x01		R1
0x02		R2
0x03		R3
0x04		R4
0x05		R5
0x06		R6

•
•
•

0x1A	XL	R26
0x1B	XH	R27
0x1C	YL	R28
0x1D	YH	R29
0x1E	ZL	R30
0x1F	ZH	R31

Contoh 3:

`LDI R15,150`

 **ERROR**

AVR Register

0x00		R0
0x01		R1
0x02		R2
0x03		R3
0x04		R4
0x05		R5
0x06		R6

•
•
•

0x1A	XL	R26
0x1B	XH	R27
0x1C	YL	R28
0x1D	YH	R29
0x1E	ZL	R30
0x1F	ZH	R31

Contoh 3:

`LDI R15,150`  **ERROR**

Hanya **R16-R31** yang dapat dioperasikan dengan instruksi yang mengandung konstanta (immediate), seperti:

- LDI Rx,K
- ANDI Rx, K
- CPI Rx, K
- SBCI
- SUBI

Penggunaan Register

- Register dapat dinamai ulang dengan menggunakan *syntax* **.DEF**
 - **Contoh:** `.def reg1=r17`
- Jika Anda membutuhkan akses pointer ke Memory, sediakan **R26** sampai **R31** untuk kebutuhan tersebut [Akan dijelaskan pada materi **Memory & Addressing**]
- Jika Anda membutuhkan untuk membaca dari program memory, gunakan **Z (R31:R30)** dan **R0** untuk kebutuhan tersebut
- 16-bit counter sebaiknya diletakkan di **R25:R24**[Akan dijelaskan pada materi **Interrupt**]

Perintah AVR

- Dapat memiliki 2 operands, 1 operand, atau tidak terdapat operand sama sekali
- Contoh:
 - Penjumlahan: ADD R1, R2 (2 Operands)
 - Negasi 2's: NEG R5 (1 Operand)
 - Load Program Memory: LPM (0 Operand)

Perintah AVR

Mnemonic	Description	Mnemonic	Description	Mnemonic	Description
Flow Control		Bit Manipulation		Load/Store	
JMP ◆	Jump absolute (24-bit)	SEC/CLC	Set/clear C flag (carry)	MOV	Copy register to register
RJMP	Branch relative (12-bit)	SEH/CLH	Set/clear H flag (half carry)	LD	Load indirect through X/Y/Z
IJMP ●	Jump indirect (Z)	SEN/CLN	Set/clear N flag (negative)	LD ●	Load indirect with postincrement
RCALL	Call subroutine	SEZ/CLZ	Set/clear Z flag (zero)	LD ●	Load indirect with predecrement
ICALL ●	Call subroutine indirect (Z)	SEI/CLI	Set/clear I flag (interrupt)	LDD ●	Load indirect with 6-bit offset
RET/RETI	Return/from interrupt	SES/CLS	Set/clear S flag (sign)	LDI	Load 8-bit immediate
CP/CPC	Compare/with carry	SEV/CLV	Set/clear V flag (overflow)	LDS ●	Load from 16-bit address
CPI	Compare with 8-bit immediate	SET/CLT	Set/clear T bit	LPS ●	Load from program space
CPSE	Compare, skip if equal	SBR/CBR	Set/clear bit in register	ST	Store indirect through X/Y/Z
SBRS/SBRC	Skip if register bit set/clear	BSET/BCLR	Set/clear bit in status register	ST ●	Store indirect with postincrement
SBIS/SBIC	Skip if I/O bit set/clear	SER/CLR	Set/clear entire register	ST ●	Store indirect with predecrement
BRcc	Conditional branch	SBI/CBI	Set/clear bit in I/O space	STD ●	Store indirect with 6-bit offset
Logical		Arithmetic		STS ●	Store to 16-bit address
AND	Logical AND	ADD/ADC	Add/with carry	IN/OUT	Input/output to/from I/O space
ANDI	Logical AND 8-bit immediate	ADIW ●	Add 6-bit immediate	PUSH/POP	Push/pop stack element
OR	Logical OR	SUB/SUBC	Subtract/with borrow	BLD/BST	Load/store T bit
ORI	Logical OR 8-bit immediate	SBIW ●	Subtract 6-bit immediate	Miscellaneous	
EOR	Logical exclusive-OR	SUBI/SBCI	Subtract 8-bit imm/w borrow	NOP	No operation
LSL/LSR	Logical shift left/right by 1 bit	INC/DEC	Increment/decrement register	SLEEP	Wait for interrupt
ROL/ROR	Rotate left/right by 1 bit	MUL ◆	Multiply $8 \times 8 \rightarrow 16$	WDR	Watchdog reset
ASR	Arithmetic shift right by 1 bit				
COM/NEG	One's/two's complement				
SWAP	Swap nibbles		Can use R16–R31 only	●	Not available on 90S1200, 1220
TST	Test for zero or minus		Can use R24–R31 only	◆	Future enhancement

Perintah AVR

- Detail dapat dilihat di dokumen **AVR Instruction**

ADD – Add without Carry

Description:

Adds two registers without the C Flag and places the result in the destination register Rd.

(i) **Operation:**
Rd ← Rd + Rr

Operasi yang dilakukan oleh perintah ADD

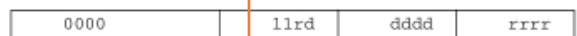
(i) **Syntax:**
ADD Rd,Rr

Operands:
 $0 \leq d \leq 31, 0 \leq r \leq 31$

Program Counter:
PC ← PC + 1

Pengaruh ke Program Counter

16-bit Opcode:



Register yang dapat digunakan dalam operasi

Syntax Perintah

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

Pengaruh ke **Status Register**

- H:** $Rd3 \cdot Rr3 + Rr3 \cdot \overline{Rd3} + \overline{Rd3} \cdot \overline{Rr3}$
Set if there was a carry from bit 3; cleared otherwise
- S:** $N \oplus V$, For signed tests.
- V:** $Rd7 \cdot Rr7 \cdot \overline{Rd7} + \overline{Rd7} \cdot Rr7 \cdot R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.
- N:** R7
Set if MSB of the result is set; cleared otherwise.
- Z:** $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
Set if the result is \$00; cleared otherwise.
- C:** $Rd7 \cdot Rr7 + Rr7 \cdot \overline{Rd7} + \overline{Rd7} \cdot Rr7$
Set if there was carry from the MSB of the result; cleared otherwise.

Contoh Program

```
.include "m8515def.inc"
```

```
forever:
```

```
    ldi    R26, 04
```

```
    ldi    R27, $01
```

```
    ld     R16, X
```

```
    andi   R16, $63
```

```
    st     X, R16
```

```
    rjmp   forever
```

AVR Assembler

The screenshot displays the AVR Assembler interface with several windows:

- Processor:** Shows system parameters: Program Counter (0x000000), Stack Pointer (0x0000), X pointer (0x0000), Y pointer (0x0000), Z pointer (0x0000), Cycle Counter (0), Frequency (4.0000 MHz), Stop Watch (0.00 us), and SREG (bits 1, 2, 4, 5, 6, 7, 8).
- Registers:** A table showing the state of all 32 registers (R0-R31). R16 and R27 are highlighted in red, both containing 0x00.
- Code Editor:** Contains assembly code for a loop labeled 'forever'. A yellow arrow points to the first instruction. Blue arrows connect the instructions to the memory window: 'ldi R26, 04' to 000000, 'ldi R27, \$02' to 000001, 'ld R16, X' to 000002, and 'andi R16, \$63' to 000003.
- Memory:** Shows the program memory layout from address 000000 to 00000E. The first four bytes (000000-000003) are highlighted in blue, corresponding to the instructions in the code editor.

```
Processor
┌──────────┴──────────┐
│ Program Counter  0x000000 │
│ Stack Pointer    0x0000   │
│ X pointer        0x0000   │
│ Y pointer        0x0000   │
│ Z pointer        0x0000   │
│ Cycle Counter    0        │
│ Frequency        4.0000 MHz │
│ Stop Watch       0.00 us  │
│ SREG             [1][2][4][5][6][7][8] │
└──────────┬──────────┘
Registers
┌──────────┴──────────┐
│ Register │
├──────────┴──────────┘
R00= 0x00  R01= 0x00
R02= 0x00  R03= 0x00
R04= 0x00  R05= 0x00
R06= 0x00  R07= 0x00
R08= 0x00  R09= 0x00
R10= 0x00  R11= 0x00
R12= 0x00  R13= 0x00
R14= 0x00  R15= 0x00
R16= 0x00  R17= 0x00
R18= 0x00  R19= 0x00
R20= 0x00  R21= 0x00
R22= 0x00  R23= 0x00
R24= 0x00  R25= 0x00
R26= 0x00  R27= 0x00
R28= 0x00  R29= 0x00
R30= 0x00  R31= 0x00

include "8515def.inc"

forever:
  ldi    R26, 04
  ldi    R27, $02
  ld     R16, X
  andi   R16, $63
  st     X, R16

  rjmp   forever

Memory
┌──────────┴──────────┐
│ Program 8/16 │
├──────────┴──────────┘
000000 A4E0
000001 B2E0
000002 0C91
000003 0376
000004 0C93
000005 FACF
000006 FFFF
000007 FFFF
000008 FFFF
000009 FFFF
00000A FFFF
00000B FFFF
00000C FFFF
00000D FFFF
00000E FFFF
e.as
```

Contoh Program

```
.include "m8515def.inc"
```

```
forever:
```

				Address
				↓
ldi	R26, 04	→	E0A4 = 1110 0000 1010 0100	0x00
ldi	R27, \$02	→	E0B2 = 1110 0000 1011 0010	0x01
ld	R16, X	→	910C = 1001 0001 0000 1100	0x02
andi	R16, \$63	→	7603 = 0111 0110 0000 0011	0x03
st	X, R16	→	930C = 1001 0011 0000 1100	0x04
rjmp	forever	→	CFFA = 1100 1111 1111 1010	0x05

AVR Operation Code (Opcode)

- AVR memiliki format instruksi yang berbeda dari MIPS
- Setiap instruksi pada AVR dapat memiliki format Opcode yang berbeda

Contoh: AVR Opcode

ADD – Add without Carry

Description:

Adds two registers without the C Flag and places the result in the destination register Rd.

Operation:
(i) $Rd \leftarrow Rd + Rr$

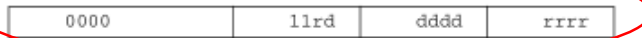
Syntax: ADD Rd,Rr
Operands: $0 \leq d \leq 31, 0 \leq r \leq 31$

Program Counter:
 $PC \leftarrow PC + 1$

• Contoh:

ADD R1, R5

16-bit Opcode:



Format opcode operasi
ADD

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

H: $Rd3 \cdot Rr3 + Rr3 \cdot \overline{R3} + \overline{R3} \cdot Rd3$
Set if there was a carry from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \cdot Rr7 \cdot \overline{R7} + \overline{Rd7} \cdot \overline{Rr7} \cdot R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
Set if the result is \$00; cleared otherwise.

C: $Rd7 \cdot Rr7 + Rr7 \cdot \overline{R7} + \overline{R7} \cdot Rd7$
Set if there was carry from the MSB of the result; cleared otherwise.

Opcode:

0000	0000	0001	0101
------	------	------	------