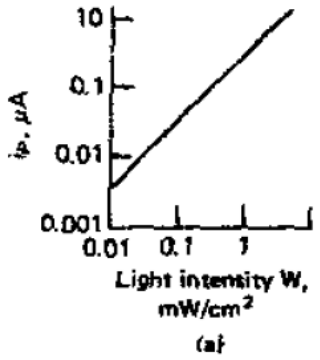
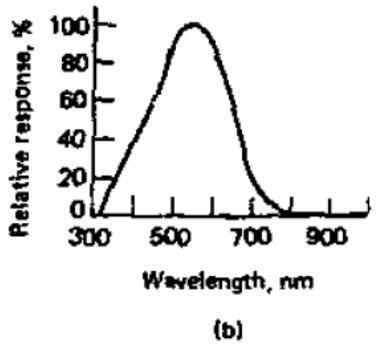
LAMPIRAN D

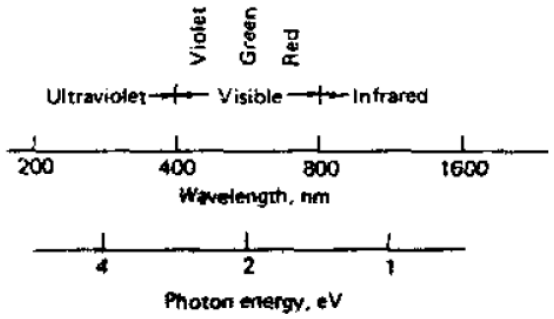
DATA SHEET

- karakteristik dioda foto
  - arus bergantung linier pada intensitas cahaya
  - respons frekuensi bergantung pada bahan (Si 900nm, GaAs 1500nm, Ge 2000nm)





- Hubungan spektrum optis dan energi
  - detektor optis umumnya menyangkut efek kuantum
  - energi foton $E_p = \frac{h}{\nu} = h\frac{c}{\lambda}$
  - frekuensifoton bergantung pada energi yang dilepas atau diterima saat elektron berpindah tingkat energinya
  - spektrum gelombang optis



ADC0801/ADC0802/ADC0803/ADC0804/ADC0805 8-Bit μP Compatible A/D Converters

General Description

The ADC0801, ADC0802, ADC0803, ADC0804 and ADC0805 are CMOS 8-bit successive approximation A/D converters that use a differential potentiometric ladder—similar to the 256R products. These converters are designed to allow operation with the NSC800 and INS8080A derivative control bus with TRI-STATE® output latches directly driving the data bus. These A/Ds appear like memory locations or I/O ports to the microprocessor and no interfacing logic is needed.

Differential analog voltage inputs allow increasing the common-mode rejection and offsetting the analog zero input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

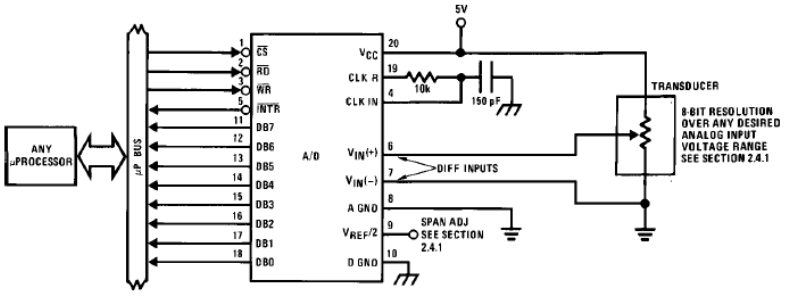
Features

- Compatible with 8080 μP derivatives—no interfacing logic needed - access time - 135 ns
- Easy interface to all microprocessors, or operates "stand alone"
- Differential analog voltage inputs
- Logic inputs and outputs meet both MOS and TTL voltage level specifications
- Works with 2.5V (LM336) voltage reference
- On-chip clock generator
- 0V to 5V analog input voltage range with single 5V supply
- No zero adjust required
- 0.3" standard width 20-pin DIP package
- 20-pin molded chip carrier or small outline package
- Operates ratiometrically or with 5 V_{DC}, 2.5 V_{DC}, or analog span adjusted voltage reference

Key Specifications

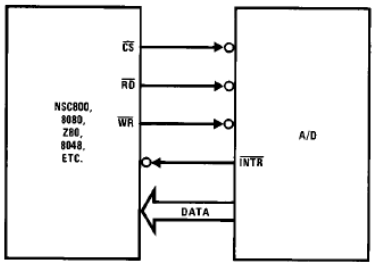
- Resolution 8 bits
- Total error ±¼ LSB, ±½ LSB and ±1 LSB
- Conversion time 100 μs

Typical Applications



TL/H/5671–1

8080 Interface



TL/H/5671–31

Error Specification (Includes Full-Scale, Zero Error, and Non-Linearity)			
Part Number	Full-Scale Adjusted	$V_{REF}/2$ = 2.500 V_{DC} (No Adjustments)	$V_{REF}/2$ = No Connection (No Adjustments)
ADC0801	±¼ LSB		
ADC0802		±½ LSB	
ADC0803	±½ LSB		
ADC0804		±1 LSB	
ADC0805			±1 LSB

Functional Description

A single point analog ground that is separate from the logic ground points should be used. The power supply bypass capacitor and the self-clocking capacitor (if used) should both be returned to digital ground. Any $V_{REF}/2$ bypass capacitors, analog input filter capacitors, or input signal shielding should be returned to the analog ground point. A test for proper grounding is to measure the zero error of the A/D converter. Zero errors in excess of ¼ LSB can usually be traced to improper board layout and wiring (see section 2.5.1 for measuring the zero error).

3.0 TESTING THE A/D CONVERTER

There are many degrees of complexity associated with testing an A/D converter. One of the simplest tests is to apply a known analog input voltage to the converter and use LEDs to display the resulting digital output code as shown in *Figure 7*.

For ease of testing, the $V_{REF}/2$ (pin 9) should be supplied with 2.560 V_{DC} and a V_{CC} supply voltage of 5.12 V_{DC} should be used. This provides an LSB value of 20 mV.

If a full-scale adjustment is to be made, an analog input voltage of 5.090 V_{DC} (5.120−1½ LSB) should be applied to the $V_{IN}(+)$ pin with the $V_{IN}(-)$ pin grounded. The value of the $V_{REF}/2$ input voltage should then be adjusted until the digital output code is just changing from 1111 1110 to 1111 1111. This value of $V_{REF}/2$ should then be used for all the tests.

The digital output LED display can be decoded by dividing the 8 bits into 2 hex characters, the 4 most significant (MS) and the 4 least significant (LS). Table I shows the fractional binary equivalent of these two 4-bit groups.
the nominal value of the digital display (when $V_{REF}/2 = 2.560V$) can be determined. For example, for an output LED display of 1011 0110 or B6 (in hex), the voltage values from the table are 3.520 + 0.120 or 3.640 V_{DC}. These voltage values represent the center-values of a perfect A/D converter. The effects of quantization error have to be accounted for in the interpretation of the test results.

For a higher speed test system, or to obtain plotted data, a digital-to-analog converter is needed for the test set-up. An accurate 10-bit DAC can serve as the precision voltage source for the A/D. Errors of the A/D under test can be expressed as either analog voltages or differences in 2 digital words.

A basic A/D tester that uses a DAC and provides the error as an analog output voltage . The 2 op amps can be eliminated if a lab DVM with a numerical subtraction feature is available to read the difference voltage, "A−C", directly. The analog input voltage can be supplied by a low frequency ramp generator and an X-Y plotter can be used to provide analog error (Y axis) versus analog input (X axis).

For operation with a microprocessor or a computer-based test system, it is more convenient to present the errors digitally. This can be done with the circuit , where the output code transitions can be detected as the 10-bit DAC is incremented. This provides ¼ LSB steps for the 8-bit A/D under test. If the results of this test are automatically plotted with the analog input on the X axis and the error (in LSB's) as the Y axis, a useful transfer function of the A/D under test results. For acceptance testing, the plot is not necessary and the testing speed can be increased by establishing internal limits on the allowed error for each code.

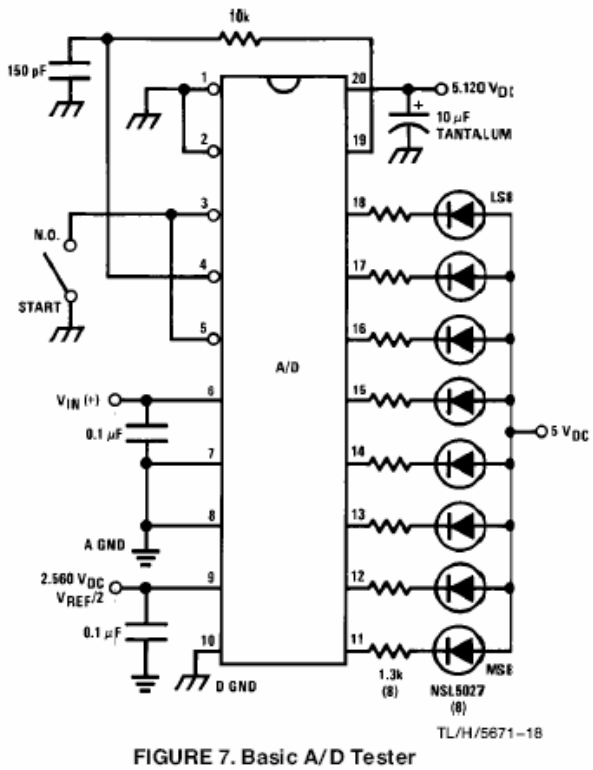


FIGURE 7. Basic A/D Tester

Features

- Compatible with MCS®-51 Products
- 4K Bytes of In-System Programmable (ISP) Flash Memory
  - Endurance: 1000 Write/Erase Cycles
- 4.0V to 5.5V Operating Range
- Fully Static Operation: 0 Hz to 33 MHz
- Three-level Program Memory Lock
- 128 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-bit Timer/Counters
- Six Interrupt Sources
- Full Duplex UART Serial Channel
- Low-power Idle and Power-down Modes
- Interrupt Recovery from Power-down Mode
- Watchdog Timer
- Dual Data Pointer
- Power-off Flag
- Fast Programming Time
- Flexible ISP Programming (Byte and Page Mode)
- Green (Pb/Halide-free) Packaging Option

Description

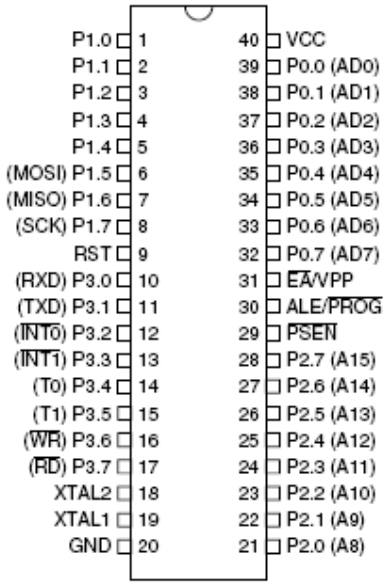
The AT89S51 is a low-power, high-performance CMOS 8-bit microcontroller with 4K bytes of In-System Programmable Flash memory. The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with In-System Programmable Flash on a monolithic chip, the Atmel AT89S51 is a powerful microcontroller which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89S51 provides the following standard features: 4K bytes of Flash, 128 bytes of RAM, 32 I/O lines, Watchdog timer, two data pointers, two 16-bit timer/counters, a five-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89S51 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next external interrupt or hardware reset.

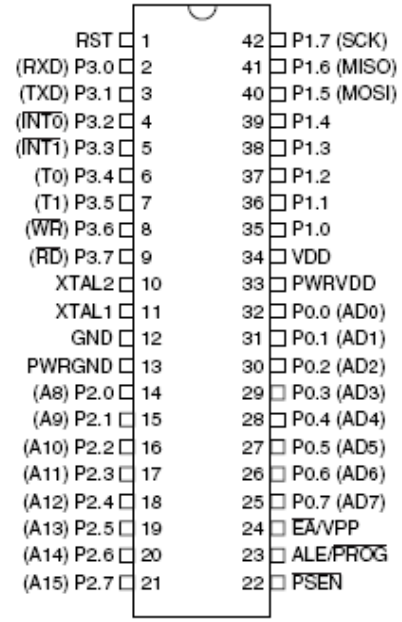
8-bit Microcontroller with 4K Bytes In-System Programmable Flash

AT89S51

40-lead PDIP



42-lead PDIP



Pin Description

VCC

Supply voltage (all packages except 42-PDIP).

GND

Ground (all packages except 42-PDIP; for 42-PDIP GND connects only the logic core and the embedded program memory).

VDD

Supply voltage for the 42-PDIP which connects only the logic core and the embedded program memory.

PWRVDD

Supply voltage for the 42-PDIP which connects only the I/O Pad Drivers. The application board **MUST** connect both VDD and PWRVDD to the board supply voltage.

PWRGND

Ground for the 42-PDIP which connects only the I/O Pad Drivers. PWRGND and GND are weakly connected through the common silicon substrate, but not through any metal link. The application board **MUST** connect both GND and PWRGND to the board ground.

Port 0

Port 0 is an 8-bit open drain bi-directional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pull-ups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. **External pull-ups are required during program verification**.

Port 1

Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pull-ups.

Port 1 also receives the low-order address bytes during Flash programming and verification.

Port Pin	Alternate Functions
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

Port 2

Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pull-ups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pull-ups.

Port 3 receives some control signals for Flash programming and verification.

Port 3 also serves the functions of various special features of the AT89S51, as shown in the following table.

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. This pin drives High for 98 oscillator periods after the Watchdog times out. The DISRTO bit in SFR AUXR (address 8EH) can be used to disable this feature. In the default state of bit DISRTO, the RESET HIGH out feature is enabled.

ALE/$\overline{PROG}$

Address Latch Enable (ALE) is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

$\overline{\text{PSEN}}$

Program Store Enable ($\overline{\text{PSEN}}$) is the read strobe to external program memory.

When the AT89S51 is executing code from external program memory, $\overline{\text{PSEN}}$ is activated twice each machine cycle, except that two $\overline{\text{PSEN}}$ activations are skipped during each access to external data memory.

$\overline{\text{EA}}$/VPP

External Access Enable. $\overline{\text{EA}}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{\text{EA}}$ will be internally latched on reset.

$\overline{\text{EA}}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier

Interrupts

The AT89S51 has a total of five interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), two timer interrupts (Timers 0 and 1), and the serial port interrupt. These interrupts are all shown in Figure 10-1.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE. IE also contains a global disable bit, EA, which disables all interrupts at once.

Note that Table 10-1 shows that bit positions IE.6 and IE.5 are unimplemented. User software should not write 1s to these bit positions, since they may be used in future AT89 products.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle.

Table 10-1. Interrupt Enable (IE) Register

(MSB) (LSB)

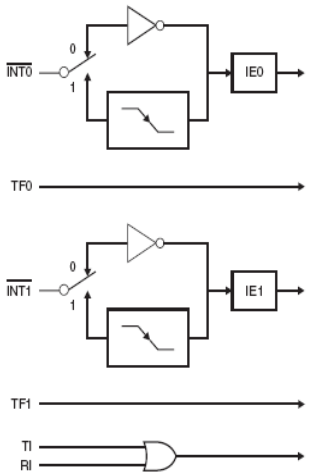
EA	–	–	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved
–	IE.5	Reserved
ES	IE.4	Serial Port interrupt enable bit
ET1	IE.3	Timer 1 interrupt enable bit
EX1	IE.2	External interrupt 1 enable bit
ET0	IE.1	Timer 0 interrupt enable bit
EX0	IE.0	External interrupt 0 enable bit
User software should never write 1s to reserved bits, because they may be used in future AT89 products.		

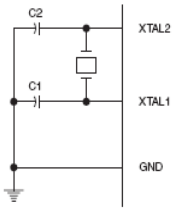
Figure 10-1. Interrupt Sources



Oscillator Characteristics

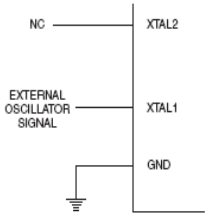
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 11-1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 11-2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Figure 11-1. Oscillator Connections



Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 11-2. External Clock Drive Configuration



Instruksi-instruksi Keluarga MCS51

A. Operasi Aritmatika

1. ADD

ADD A,Rn

Tambahkan Akumulator A dengan Rn di mana n = 0…7 dan simpan hasil di Akumulator A

Contoh:

Add A,R7

Isi dari R7 akan ditambahkan dengan akumulator A dan hasilnya disimpan di Akumulator A

ADD A,direct

Tambahkan Akumulator A dengan data di alamat memori tertentu secara langsung.

Contoh:

Add A,00H

Isi dari Akumulator A akan ditambahkan dengan isi dari memori RAM Internal di alamat 00H

ADD A,@Ri

Tambahkan Akumulator A dengan data yang berada di alamat Ri (ditunjuk oleh Ri) dan hasilnya disimpan di Akumulator A. Ri adalah Register Index di mana pada MCS51 adalah berupa R0 atau R1

Contoh:

Add A,@R0

Isi dari Akumulator A akan ditambahkan dengan isi dari memori RAM Internal yang ditunjuk oleh R0. Apabila R0 berisi 05H maka, isi dari alamat 05H akan dijumlahkan dengan Akumulator A dan hasilnya disimpan di Akumulator A

ADD A,#data

Tambahkan Akumulator A dengan sebuah konstanta dan hasilnya disimpan dalam akumulator A.

Contoh:

Add A,#05H

Isi Akumulator A ditambah dengan data 05H dan hasilnya disimpan dalam Akumulator A

2. ADDC

ADDC A,Rn

Tambahkan Akumulator A dengan Rn di mana n = 0…7 dan simpan hasil di Akumulator A

Contoh:

Addc A,R7

Isi dari R7 akan ditambahkan dengan akumulator A beserta carry flag dan hasilnya disimpan di Akumulator A. Apabila carry flag set maka hasil yang tersimpan di Akumulator A adalah A + R7 + 1.

ADDC A,direct

Tambahkan Akumulator A dan carry flag dengan data di alamat memori tertentu secara langsung.

Contoh:

Addc A,00H

Isi dari Akumulator A akan ditambahkan dengan isi dari memori RAM Internal di alamat 00H beserta carry flag dan hasilnya disimpan di Akumulator A, Apabila carry flag set maka hasil yang tersimpan di Akumulator A adalah A + isi alamat 00H + 1

ADDC A,@Ri

Tambahkan Akumulator A beserta carry flag dengan data yang berada di alamat Ri (ditunjuk oleh Ri) dan hasilnya disimpan di Akumulator A. Ri adalah Register Index di mana pada MCS51 adalah berupa R0 atau R1

ADDC A,#data

Tambahkan Akumulator A beserta carry flag dengan sebuah konstanta dan hasilnya disimpan dalam akumulator A.

Contoh:

Addc A,#05H Isi Akumulator A beserta carry flag ditambah dengan data 05H dan hasilnya disimpan dalam Akumulator A. Apabila carry flag set maka hasil di Akumulator A adalah A + 5H+ 1.

3. SUBB

SUBB A,Rn

Lakukan pengurangan data di Akumulator A dengan Rn (n = 0…7) dan simpan hasilnya di Akumulator A

Contoh:

Subb A,R0

Data di akumulator A beserta carry flagnya dikurangi dengan isi R0 dan hasilnya disimpan di Akumulator A

SUBB A,direct

Lakukan pengurangan data di Akumulator A dengan data di memori tertentu yang ditunjuk secara langsung.

Contoh:

Subb A,00H

Data di Akumulator A beserta carry flagnya dikurangi dengan data dialamat 00H dari RAM Internal dan hasilnya disimpan di Akumulator A

SUBB A,@Ri

Lakukan pengurangan data di Akumulator A beserta carry flag dengan data yang ditunjuk oleh Ri (Register Index) di mana Ri dapat berupa R0 atau R1

Contoh:

SUBB A,@R0

Data di Akumulator A beserta carry flagnya dikurangi dengan data yang ditunjuk oleh R0 dan hasilnya disimpan di Akumulator A

SUBB A,#data

Lakukan pengurangan data di Akumulator A beserta carry flag dengan sebuah konstanta dan hasilnya disimpan di Akumulator A

Contoh:

SUBB A,#05H

Data di Akumulator A beserta carry flag dikurangi dengan data 05H dan hasilnya disimpan di Akumulator A

4. INC

INC A

Tambahkan nilai Akumulator A dengan 1 dan hasilnya disimpan di Akumulator A

INC Rn

Tambahkan nilai Rn (n= 0…7) dengan 1 dan hasilnya disimpan di Rn tersebut

INC direct

Tambahkan data yang di RAM Internal yang alamatnya ditunjuk secara langsung dengan 1 dan hasilnya disimpan di alamat tersebut.

Contoh:

Inc 00H

Data di alamat 00H ditambah dengan 1 dan hasilnya disimpan di alamat 00H.

INC @Ri

Tambahkan data yang alamatnya ditunjuk oleh Ri (Register Index) dengan 1 dan simpan hasilnya di alamat tersebut.

Contoh:

Inc @R1

Data di alamat yang ditunjuk oleh R1 dan hasilnya disimpan di alamat tersebut, apabila R1 berisi 10H maka data di alamat 10H ditambah dengan 1 dan simpan kembali di alamat 10H.

INC DPTR

5. DEC

DEC A

Lakukan pengurangan pada nilai Akumulator A dengan 1 dan hasilnya disimpan di Akumulator A

DEC Rn

Lakukan pengurangan pada nilai Rn (n= 0…7) dengan 1 dan hasilnya disimpan di Rn tersebut

DEC direct

Lakukan pengurangan pada data yang di RAM Internal yang alamatnya ditunjuk secara langsung dengan 1 dan hasilnya disimpan di alamat tersebut.

Contoh:

Dec 00H

Data di alamat 00H dikurangi dengan 1 dan hasilnya disimpan di alamat 00H.

DEC @Ri

Lakukan pengurangan pada data yang alamatnya ditunjuk oleh Ri (Register Index) dengan 1 dan simpan hasilnya di alamat tersebut.

Contoh:

DEC @R1

Data di alamat yang ditunjuk oleh R1 dan hasilnya disimpan di alamat tersebut, apabila R1 berisi 10H maka data di alamat 10H dikurangi dengan 1 dan simpan kembali di

alamat 10H.

B. Operasi Logika dan Manipulasi Bit

1. ANL

ANL A,Rn

Melakukan operasi AND antara akumulator A dan Rn (R0…R7) dan hasilnya disimpan di akumulator A

ANL A,direct

Melakukan operasi AND antara akumulator A dan alamat langsung dan hasilnya disimpan di akumulator A.

Contoh:

ANL A,05H

Akumulator A di AND dengan data di alamat 05H dan hasilnya disimpan di akumulator A

ANL A,@Ri

Melakukan operasi AND antara akumulator A dan data yang ditunjuk oleh Register Index (R0 atau R1) serta hasilnya disimpan di akumulator A.

Contoh:

ANL A,@R0

Akumulator A di AND dengan data yang ditunjuk oleh R0, misalkan R0 berisi 50H, maka akumulator A di AND dengan data yang tersimpan di alamat 50H dan hasilnya disimpan di akumulator A.

ANL A,#data

Melakukan operasi AND antara akumulator A dan immediate data serta hasilnya disimpan di akumulator A

ANL direct,A

Melakukan operasi AND antara alamat langsung dengan akumulator A serta hasilnya disimpan di alamat langsung tersebut.

Contoh:

ANL 07H,A

Data di alamat 07H di AND dengan akumulator A dan hasilnya kembali disimpan di alamat 07H

ANL direct,#data

Melakukan operasi AND antara alamat langsung dengan immediate data serta hasilnya disimpan di alamat langsung tersebut.

2. ORL

ORL A,Rn

Melakukan operasi OR antara akumulator A dan Rn (R0…R7) dan hasilnya disimpan di akumulator A

ORL A,direct

Melakukan operasi OR antara akumulator A dan alamat langsung dan hasilnya disimpan di akumulator A.

Contoh:

ORL A,05H

Akumulator A di OR dengan data di alamat 05H dan hasilnya disimpan di akumulator A

ORL A,@Ri

Melakukan operasi OR antara akumulator A dan data yang ditunjuk oleh Register Index (R0 atau R1) serta hasilnya disimpan di akumulator A.

Contoh:

ORL A,@R0

Akumulator A di OR dengan data yang ditunjuk oleh R0, misalkan R0 berisi 50H, maka akumulator A di OR dengan data yang tersimpan di alamat 50H dan hasilnya disimpan di akumulator A.

ORL A,#data

Melakukan operasi OR antara akumulator A dan immediate data serta hasilnya disimpan di akumulator A

ORL direct,A

Melakukan operasi OR antara alamat langsung dengan akumulator A serta hasilnya disimpan di alamat langsung tersebut.

Contoh:

ORL 07H,A

Data di alamat 07H di OR dengan akumulator A dan hasilnya kembali disimpan di alamat 07H

ORL direct,#data

Melakukan operasi OR antara akumulator A dan immediate data serta hasilnya disimpan di akumulator A

3. CLR

CLR A

Memberikan nilai 0 pada 8 bit Akumulator A

4. CPL

CPL A

Melakukan komplemen pada setiap bit dalam akumulator A.

Contoh :

Bila nilai akumulator A adalah 55H atau 01010101b, maka setelah terjadi proses komplemen nilai akumulator A berubah menjadi AAH atau 10101010b.

5. RL

RL A

Melakukan pergeseran ke kiri 1 bit untuk setiap bit dalam akumulator A

Contoh:

Nilai Akumulator A adalah 05H atau 00000101b, setelah dilakukan proses pergeseran maka nilai Akumulator A akan berubah menjadi 00001010b atau 0AH.

6. RR

RR A

Melakukan pergeseran ke kanan 1 bit untuk setiap bit dalam akumulator A

Contoh:

Nilai Akumulator A adalah 05H atau 00000101b, setelah dilakukan proses pergeseran maka nilai Akumulator A akan berubah menjadi 10000010b atau 0AH.

7. SWAP

SWAP A

Melakukan operasi penukaran nibble tinggi dan nibble rendah di akumulator A

Contoh:

Isi akumulator A adalah 51H, setelah instruksi SWAP A dilakukan maka data 5 di nibble tinggi akan ditukar dengan data 1 di nibble rendah menadi 15H

8. SETB

SETB bit

Set bit atau mengubah bit-bit pada RAM Internal maupun register yang dapat dialamat secara bit (bit addressable) menjadi 1

9. JC

JC rel

Melakukan lompatan ke suatu alamat yang didefinisikan apabila carry flag set. Apabila carry flag clear maka program akan menjalankan instruksi selanjutnya.

Contoh:

```
Jc Alamat1
Mov A,#05H
Alamat1: Mov R1,#00H
```

Apabila carry flag set, maka program akan lompat label alamat 1 dan menjalankan instruksi Mov R1,#00H, namun bila carry flag clear maka program akan menjalankan instruksi Mov A,#05H terlebih dahulu sebelum menjalankan instruksi di label alamat 1.

10 JNC

JNC rel

Melakukan lompatan ke suatu alamat yang didefinisikan apabila carry flag clear. Apabila carry flag set maka program akan menjalankan instruksi selanjutnya.

Contoh:

Jnc Alamat1

Mov A,#05H

Alamat1: Mov R1,#00H

Apabila carry flag clear, maka program akan lompat label alamat 1 dan menjalankan instruksi Mov R1,#00H, namun bila carry flag set maka program akan menjalankan instruksi Mov A,#05H terlebih dahulu sebelum menjalankan instruksi di label alamat 1.

C. Transfer Data

1. MOV

MOV A,Rn

Melakukan pemindahan data dari Rn (R0…R7) menuju ke akumulator A

MOV A,direct

Melakukan pemindahan data dari alamat langsung ke akumulator A

Mov A,@Ri

Melakukan pemindahan data dari alamat yang ditunjuk oleh Register Index (R0 atau R1) menuju ke akumulator A

Mov A,#data

Melakukan pemindahan data dari immediate menuju ke akumulator A

Contoh:

Data EQU 05H

Mov A,#Data

Konstanta Data yang dideklarasikan sebagai 05H dipindah ke akumulator A sehingga nilai akumulator A menjadi 05H

Mov Rn,A

Melakukan pemindahan data dari akumulator A menuju ke Rn (R0…R7)

Mov Rn,direct

Melakukan pemindahan data dari alamat langsung menuju ke Rn (R0…R7)

Contoh:

Mov R7,10H

Data di alamat 10H dipindah ke dalam R7

Mov Rn,#data

Melakukan pemindahan data dari immediate menuju ke Rn (R0…R7)

Contoh:

Mov R7,#05H

Data 05H dipindah ke dalam R7

Mov direct,A

Melakukan pemindahan data dari akumulator A menuju ke alamat langsung

Contoh:

Mov 10H,A

Data di akumulator A dipindah ke alamat 10H

Mov direct,Rn

Melakukan pemindahan data dari Rn (R0…R7) menuju ke alamat langsung

Mov direct,direct

Melakukan pemindahan data dari alamat langsung menuju ke alamat langsung.

Mov direct,@Ri

Melakukan pemindahan data dari alamat yang ditunjuk oleh Register Index (R0 atau R1) ke alamat langsung

Contoh:

Mov 05H,@R0

Bila R0 sebelumnya berisi 20H, maka nilai atau data yang tersimpan di alamat 20H akan dipindah ke alamat 05H.

Mov direct,#data

Melakukan pemindahan data dari immediate ke alamat langsung.

Mov @Ri,A

Melakukan pemindahan data dari akumulator A menuju ke alamat yang ditunjuk oleh Register Index (R0 atau R1).

Mov @Ri,direct

Melakukan pemindahan data dari alamat langsung menuju ke alamat yang ditunjuk oleh Register Index (R0 atau R1)

Mov @Ri,#data

Melakukan pemindahan data immediate menuju ke alamat yang ditunjuk oleh Register Index (R0 atau R1)

Mov DPTR,#data16

Melakukan pemindahan data immediate 16 bit menuju ke DPTR.

Contoh:

Mov DPTR,#2000H

Data 2000H dalam bentuk 16 bit dipindah ke alamat Register DPTR

Movc A,@A+DPTR

Contoh:

Mov A,#50H

Mov DPTR,#2000H

Movc A,@A+DPTR

Data yang terletak di komponen memori di luar AT89S51 dan terletak pada alamat 2000H + 50H akan dibaca dan hasilnya disimpan di akumulator A

D. Percabangan

1. ACALL

ACALL addr11

Melakukan lompatan ke suatu subroutine yang ditunjuk oleh alamat pada addr11. Lompatan yang dapat dilakukan berada di area sebesar 2K byte.

2.RET

RET

Instruksi ini digunakan pada saat kembali dari subroutine yang dipanggil dengan instruksi ACALL atau LCALL.

RETI

Instruksi ini digunakan untuk melompat ke alamat tempat akhir instruksi yang sedang dijalankan ketika.

3. JUMP

LJMP addr16

Long Jump, melompat dan menjalankan program yang berada di alamat yang ditentukan oleh addr16.

Contoh:

LJMP Lompatan2

Mov A,#05H

Lompatan2: Mov R0,#00H

Program akan melompat ke alamat lompatan 2 dan menjalankan instruksi Mov R0,#00H, tanpa melalui instruksi MOV A,#05H

JZ rel

Melakukan lompatan ke alamat yang ditentukan apabila akumulator A adalah 00H dan langsung meneruskan instruksi dibawahnya bila akumulator A tidak 00H.

Contoh:

JZ Lompat1

MOV A,#07H

Lompat1: MOV B,#00H

Apabila nilai akumulator A tidak 00H maka program akan langsung meneruskan instruksi dibawahnya yaitu MOV A,#07H dan program akan menjalankan instruksi di alamat Lompat1 yaitu MOV B,#00H apabila nilai akumulator A adalah 00H.

JNZ rel

Melakukan lompatan ke alamat yang ditentukan apabila akumulator A adalah bukan 00H dan langsung meneruskan instruksi dibawahnya bila akumulator A adalah 00H.

Contoh:

JNZ Lompat1

MOV A,#07H

Lompat1: MOV B,#00H

Apabila nilai akumulator A adalah 00H maka program akan langsung meneruskan instruksi dibawahnya yaitu MOV A,#07H dan program akan menjalankan instruksi di alamat Lompat1 yaitu MOV B,#00H apabila nilai akumulator A adalah bukan 00H.

4. CJNE

Instruksi ini melakukan perbandingan antara data tujuan dan data sumber serta melakukan lompatan ke alamat yang ditentukan apabila hasil perbandingan tidak sama.

CJNE A,#data,rel

Melakukan perbandingan antara akumulator A dan data immediate serta melakukan lompatan ke alamat yang ditentukan apabila hasil perbandingan tidak sama.

Contoh:

CJNE A,#00H,lompat1

Program akan menuju ke alamat lompat 1 apabila data akumulator A tidak sama dengan data 00H..

5. DJNZ

DJNZ Rn,rel

Melakukan pengurangan pada Rn (R0…R7) dengan 1 dan lompat ke alamat yang ditentukan apabila hasilnya bukan 00.

Apabila hasilnya telah mencapai 00, maka program akan terus menjalankan instruksi di bawahnya.

Contoh:

Tunggu: DJNZ R7,Tunggu

RET

Selalu melakukan lompatan ke alamat tunggu dan mengurangi R7 dengan 1 selama nilai R7 belum mencapai 00

6. NOP

NOP

Instruksi ini berfungsi untuk melakukan tundaan pada program sebesar 1 cycle tanpa mempengaruhi register-register maupun flag.