I. PENDAHULUAN

Jika kita bicara tentang mikrokontroler maka tidak terlepas dengan pengertian atau definisi tentang komputer. Mengapa? Karena ada kesamaan-kesamaan antara mikrokontroler dengan komputer (atau mikrokomputer), antara lain:

- Sama-sama memiliki unit pengolah pusat atau yang lebih dikenal dengan CPU (*Central Processing Unit*);
- CPU tersebut sama-sama menjalankan program dari suatu lokasi atau tempat, biasanya dari ROM (*Read Only Memory*) atau RAM (*Random Access Memory*);
- Sama-sama memiliki ROM, hanya saja pada komputer digunakan untuk menyimpan program BIOS (*Basic Input Outsput System*), sedangkan pada mikrokontroler (yang dikenal dengan Flash PEROM) digunakan untuk menyimpan program yang akan dijalankan mikrokontroler yang bersangkutan (sering dinamakan sebagai *firmware*);
- Sama-sama memiliki RAM yang digunakan untuk menyimpan data-data sementara atau yang lebih dikenal dengan variabel-variabel;
- Sama-sama memiliki beberapa keluaran dan masukan yang digunakan untuk melakukan komunikasi timbal-balik dengan dunia luar.

Apa yang membedakan antara mikrokontroler dengan mikrokomputer?

Begitu mungkin pertanyaan yang ada di benak kita, saat kita membaca beberapa daftar kesamaan tersebut. Sama sekali berbeda, itu jawaban yang penulis berikan, karena **mikrokontroler adalah versi mini dan untuk aplikasi khusus dari mikrokomputer!**

Berikut penulis berikan kembali daftar kesamaan yang pernah ditulis sebelumnya dengan menekankan pada perbedaan antara mikrokontroler dan mikrokomputer:

- CPU pada Mikrokomputer berada eksternal dalam suatu sistem, sampai saat ini kecepatan operasionalnya sudah mencapai tingkat lebih dari 2

GHz, sedangkan CPU pada mikrokontroler berada internal dalam sebuah *chip*, kecepatan bekerja masih cukup rendah, dalam orde MHz (misalnya, 24 MHz, 40 MHz dan lain sebagainya). Kecepatan yang relatif rendah ini sudah mencukupi untuk aplikasi-aplikasi berbasis mikrokontroler

- Jika CPU pada mikrokomputer menjalankan program dalam ROM atau yang lebih dikenal dengan BIOS pada saat awal dihidupkan, kemudian mengambil atau menjalankan program yang tersimpan dalam *hard disk*, sedangkan mikrokontroler sejak awal menjalankan program yang tersimpan dalam ROM internal-nya (bisa berupa Mask ROM atau Flash PEROM). Sifat memori program ini *non volatile*, artinya tetap akan tersimpan walaupun tidak diberi catu daya
- RAM pada mikrokomputer bisa mencapai ukuran sekian MByte dan bisa di-*upgrade* ke ukuran yang lebih besar dan berlokasi di luar chip CPU-nya, sedangkan RAM pada mikrokontroler ada di dalam chip mikrokontroler yang bersangkutan dan ukurannya sangat minim, misalnya 128 byte, 256 byte dan seterusnya. Ukuran yang relatif kecil inipun dirasa cukup untuk aplikasi-aplikasi mikrokontroler
- Keluaran dan masukan pada mikrokomputer jauh lebih kompleks dibandingkan dengan mikrokontroler. Selain itu, pada mikrokontroler tingkat akses keluaran dan masukan bisa dalam satuan per bit
- Jika diamati lebih lanjut, bisa dikatakan bahwa mikrokomputer merupakan komputer serbaguna atau *general purpose computer*, bisa dimanfaatkan untuk berbagai macam aplikasi (atau perangkat lunak), sedangkan mikrokontroler adalah *special purpose computer* atau komputer untuk tujuan khusus, hanya satu macam aplikasi saja.

Ciri khas mikrokontroler lainnya, antara lain:

- ‘Tertanam’ (atau *embedded*) dalam beberapa piranti (umumnya merupakan produk konsumen) atau yang dikenal dengan istilah *embedded system* atau *embedded controller*
- Terdedikasi untuk satu macam aplikasi saja

- Hanya membutuhkan daya yang rendah (*low power*) sekitar 50 mWatt (bandingkan dengan komputer yang bisa mencapai 50 Watt lebih)
- Memiliki beberapa keluaran maupun masukan yang terdedikasi, untuk tujuan atau fungsi-fungsi khusus
- Kecil dan relatif lebih murah
- Seringkali tahan-banting, terutama untuk aplikasi-aplikasi yang berhubungan dengan mesin atau otomotif atau militer.

Sejarah Singkat Mikrokontroler

Mikrokontroler populer yang pertama dibuat oleh Intel pada tahun 1976, yaitu mikrokontroler 8-bit Intel 8748. Mikrokontroler tersebut adalah bagian dari keluarga mikrokontroler MCS-48. Sebelumnya, Texas instruments telah memasarkan mikrokontroler 4-bit pertama yaitu TMS 1000 pada tahun 1974. TMS 1000 yang mulai dibuat sejak 1971 adalah mikrokomputer dalam sebuah chip, lengkap dengan RAM dan ROM.

Jenis-jenis Mikrokontroler

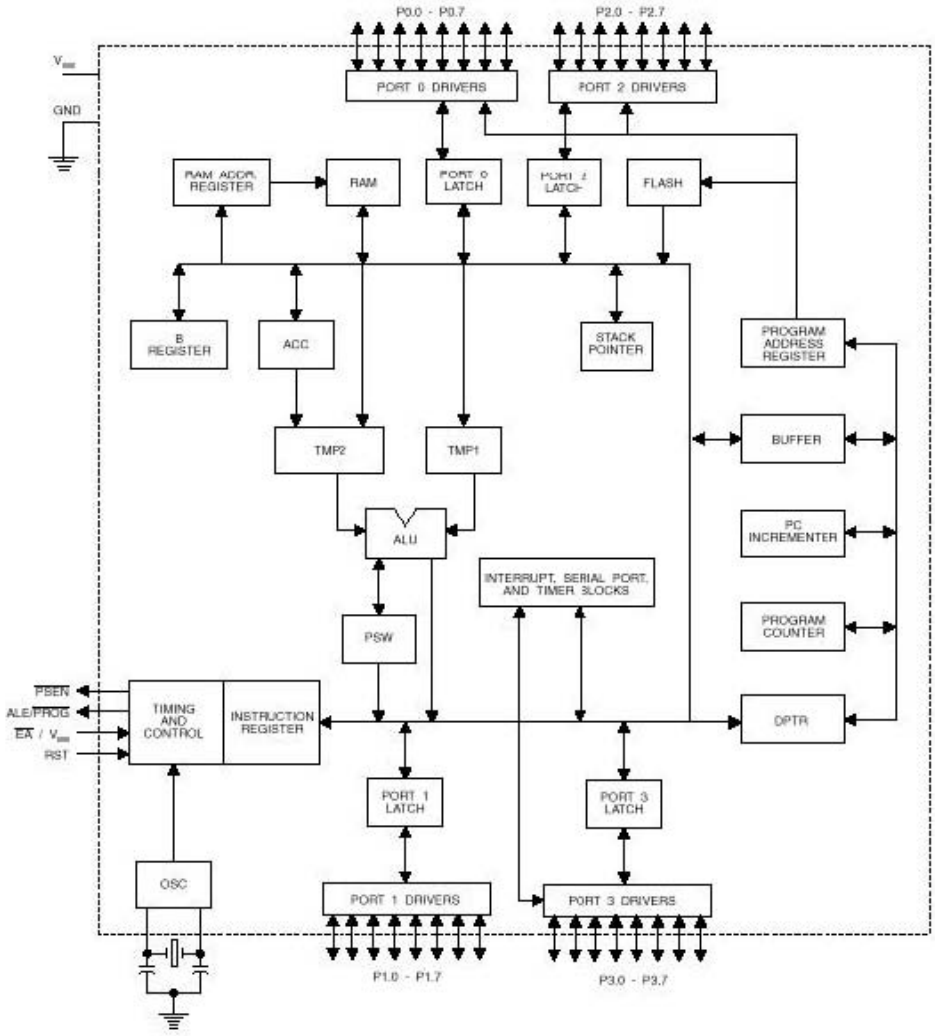
Mikrokontroler yang beredar saat ini dapat dibedakan menjadi dua macam, berdasarkan arsitekturnya:

- Tipe **CISC** atau *Complex Instruction Set Computing* yang lebih kaya instruksi tetapi fasilitas internal secukupnya saja, misalkan seri AT89 memiliki 255 instruksi.
- Tipe **RISC** atau *Reduced Instruction Set Computing* yang justru lebih kaya fasilitas internalnya tetapi jumlah instruksi secukupnya, misalkan seri PIC16F hanya ada sekitar 30-an instruksi

Fasilitas internal yang penulis maksudkan di sini antara lain: jumlah dan macam register internal, pewaktu dan/atau pencacah, ADC atau DAC, unit komparator, interupsi eksternal maupun internal dan lain sebagainya.

II. MIKROKONTROLLER AT89S51

Mikrokontroler 89S51 merupakan mikrokomputer CMOS 8 bit dengan 4 Kbytes *Flash Programmable Memory*. Arsitektur 89S51 ditunjukkan pada gambar 1.



Gambar 1. Blok Diagram 89S51

A. FITUR

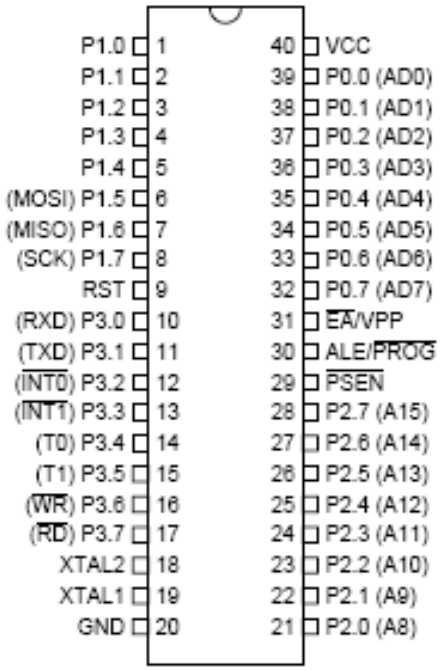
- Kompatibel dengan produk MCS-51
- 4K byte In System Programmable Flas Memory. Dapat dilakukan pemrograman 1000 tulis dan hapus
- Range catu daya 4,0 V s/d 5,0 V
- Operasi statis: 0 Hz s/d 33 MHz
- Tiga Tingkat Program memory lock
- 128 x 8 bit RAM internal

- 32 Programmable Jalur I/O
- Dua 16 bit Timer/ Counter
- Enam Sumber Interupsi
- Full Duplex Serial Channel
- Low Power Idle dan Mode Power Down
- Watcht Dog Timer
- Dua Data Pointer
- Power Off Flag
- Fast Programming Time
- Fleksibel ISP programming

B. DISKRIPSI

AT89S51 mempunyai konsumsi daya rendah. Mikrokontroller 8-bit CMOS ini mempunyai 4K byte momori Flash ISP (*In System Programmable*, dapat diprogram di dalam sistem). Divais ini dibuat dengan teknologi memori *non volatile* kerapatan tinggi dan kompatibel dengan standar industri 8051, set instruksi dan pin keluaran. Flash yang berada didalam chip memungkinkan memori program untuk diprogram ulang pada saat chip didalam sistem atau dengan menggunakan programmer memori *non volatile* konvensional. Dengan mengkombinasikan CPU 8 bit yang serbaguna dengan flash ISP pada chip, ATMEL 89S51 merupakan mikrokontroller yang luar biasa yang memberikan fleksibelitas yang tinggi dan penyelesaian biaya yang efektif untuk beberapa aplikasi kontrol.

AT89S51 memberikan fitur-fitur standar sebagai berikut: 4K byte Flash, 128 byte RAM, 32 jalur I/O, *Timer Wachtdog*, dua data pointer, dua 16 bit timer/ *counter*, lima vektor interupsi dua level, sebuah port serial *full dupleks*, oscilator internal, dan rangkaian clock. Selain itu AT89S51 didisain dengan logika statis untuk operasi dengan frekuensi sampai 0 Hz dan didukung dengan mode penghematan daya. Pada mode *idle* akan menghentikan CPU sementara RAM, timer/ counter, serial port dan sistem interupsi tetap berfungsi. Mode Power Down akan tetap menyimpan isi dari RAM tetapi akan membekukan osilator, menggagalkan semua fungsi chip sampai interupsi eksternal atau reset hardware ditemui.



Gambar 2. Konfigurasi PIN AT89S51

C. DISKRIPSI PIN

VCC Tegangan Supply

GND Ground

Port 0

Port 0, merupakan port I/O 8 bit *open drain* dua arah. Sebagai sebuah port, setiap pin dapat mengendalikan 8 input TTL. Ketika logika “1” dituliskan ke port 0, maka port dapat digunakan sebagai input dengan impedansi tinggi.Port 0 dapat juga dikonfigurasikan untuk multipleksing dengan address/ data bus selama mengakses memori program atau data eksternal. Pada mode ini P0 harus mempunyai *pull up*

Port 1

Port 1 merupakan port I/0 8 bit dua arah dengan *internal pull up*. *Buffer* output port 1 dapat mengendalikan empat TTL input. Ketika logika “1” dituliskan ke port 1, maka port ini akan mendapatkan *internal pull up* dan dapat digunakan sebagai input. Port 1 juga menerima alamat byte rendah selama pemrograman dan verifikasi Flash

Port Pin Fungsi Alternatif

P1.5 MOSI (digunakan untuk *In System Programming*)

P1.6 MISO (digunakan untuk *In System Programming*)

P1.7 SCK (digunakan untuk *In System Programming*)

Port 2

Port 2 merupakan port I/O 8 bit dua arah dengan *internal pull up*. *Buffer* output port 2 dapat mengendalikan empat TTL input. Ketika logika “1” dituliskan ke port 2, maka port ini akan mendapatkan *internal pull up* dan dapat digunakan sebagai input.

Port 3

Port 3 merupakan port I/O 8 bit dua arah dengan *internal pull up. Buffer* output port 3 dapat mengendalikan empat TTL input. Ketika logika “1” dituliskan ke port 3, maka port ini akan mendapatkan *internal pull up* dan dapat digunakan sebagai input. Port 3 juga melayani berbagai macam fitur khusus, sebagaimana yang ditunjukkan pada tabel berikut:

Tabel 1. Port 3 yang melayani fitur khusus

Port Pin	Fungsi Alternatif
P3.0	RXD (port serial input)
P3.1	TXD (port serial output)
P3.2	INT0 (interupsi eksternal 0)
P3.3	INT1 (interupsi eksternal 1)
P3.4	T0 (input eksternal timer 0)
P3.5	T1 (input eksternal timer 1)
P3.6	WR (write strobe memori data eksternal)
P3.7	WR (read strobe memori program eksternal)

RST

Input Reset. Logika high “1” pada pin ini untuk dua siklus mesin sementara oscilator bekerja maka akan mereset devais.

ALE/ PROG

Address Latch Enable (ALE) merupakan suatu pulsa output untuk mengunci byte low dari alamat selama mengakses memori eksternal. Pin ini juga merupakan input pulsa pemrograman selama pemrograman flash (paralel). Pada operasi normal, ALE mengeluarkan suatu laju konstan 1/6 dari frekuensi oscilator dan dapat digunakan untuk pewaktu eksternal.

PSEN

Program Store Enable merupakan *strobe read* untuk memori program eksternal.

EA/ VPP

Eksternal Access Enable. EA harus di hubungkan ke GND untuk enable devais, untuk memasuki memori program eksternal mulai alamat 0000H s/d FFFFH. EA harus dihubungkan ke VCC untuk akses memori program internal. Pin ini juga menerima tegangan pemrogramman (VPP) selama pemrograman Flash

XTAL1

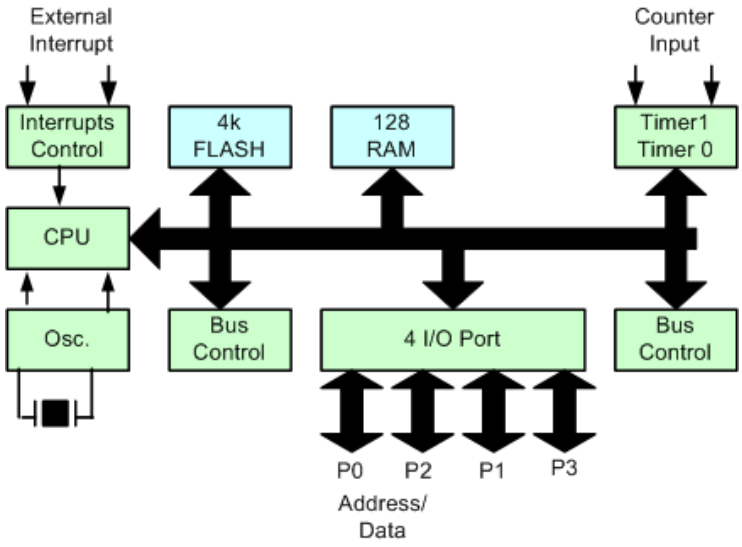
Input untuk penguat oscilator inverting dan input untuk rangkaian internal clock

XTAL2

Output dari penguat oscilator inverting.

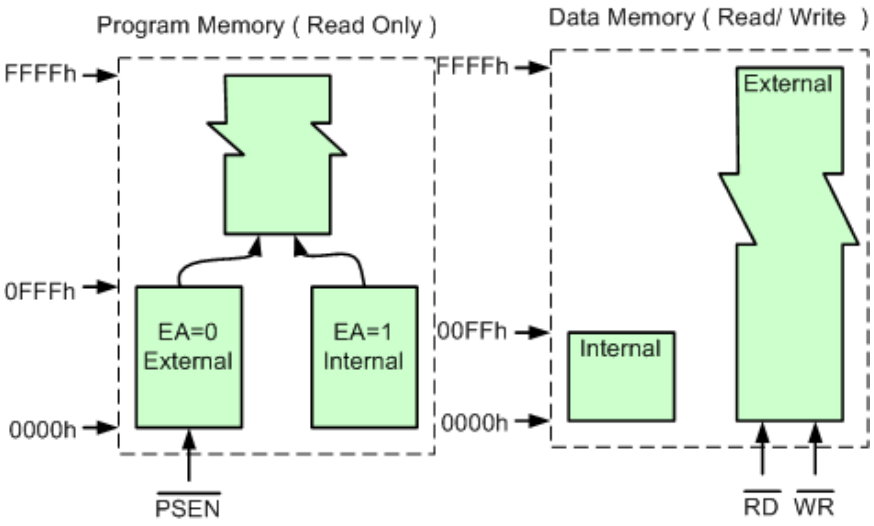
D. ORGANISASI MEMORI

Semua divais 8051 mempunyai ruang alamat yang terpisah untuk memori program dan memori data, seperti yang ditunjukkan pada Gambar 3 dan Gambar 4. Pemisahan secara logika dari memori program dan data, mengijinkan memori data untuk diakses dengan pengalamatan 8 bit, yang dengan cepat dapat disimpan dan dimanipulasi dengan CPU 8 bit. Selain itu, pengalamatan memori data 16 bit dapat juga dibangkitkan melalui register DPTR. Memori program (ROM, EPROM dan FLASH) hanya dapat dibaca, tidak ditulis. Memori program dapat mencapai sampai 64K byte. Pada 89S51, 4K byte memori program terdapat didalam chip. Untuk membaca memori program eksternal mikrokontroller mengirim sinyal PSEN (*Program Store ENable*)



Gambar 3. Diagram blok mikrokontroller 8051

Memori data (RAM) menempati ruang alamat yang terpisah dari memori program. Pada keluarga 8051, 128 byte terendah dari memori data, berada di dalam chip. RAM eksternal (maksimal 64K byte). Dalam pengaksesan RAM Eksternal, mikrokontroler mengirimkan sinyal RD (baca) dan WR (tulis).

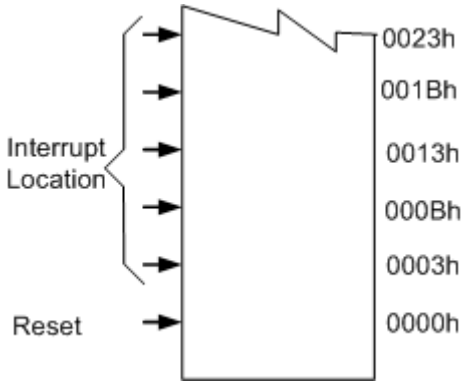


Gambar 4. Arsitektur Memori Mikrokontroller 8051

1. Program Memory

Gambar 4 menunjukkan suatu peta bagian bawah dari memori program. Setelah reset CPU mulai melakukan eksekusi dari lokasi 0000H. Sebagaimana yang ditunjukkan pada Gambar 4, setiap interupsi ditempatkan pada suatu

lokasi tertentu pada memori program. Interupsi menyebabkan CPU untuk melompat ke lokasi dimana harus dilakukan suatu layanan tertentu. Interupsi Eksternal 0, sebagi contoh, menempati lokasi 0003H. Jika Interupsi Eksternal 0 akan digunakan, maka layanan rutin harus dimulai pada lokasi 0003H. Jika interupsi ini tidak digunakan, lokasi layanan ini dapat digunakan untuk berbagai keperluan sebagai Memori Program.



Gambar 5. Peta Interupsi mikrokontroller 8051

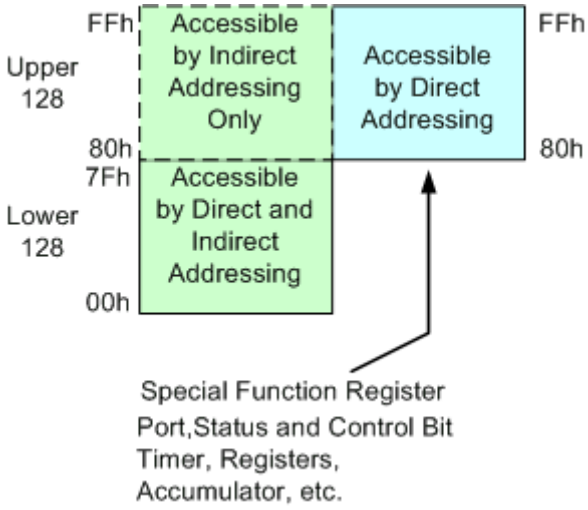
2. Memory Data

Gambar 4 menunjukkan ruang memori data internal dan eksternal pada keluarga 8051. CPU membangkitkan sinyal RD dan WR yang diperlukan selama akses RAM eksternal. Memori data internal terpetakan seperti pada gambar 4. Ruang memori dibagi menjadi tiga blok, yang diacukan sebagai 128 byte lower, 128 byte upper dan ruang SFR.

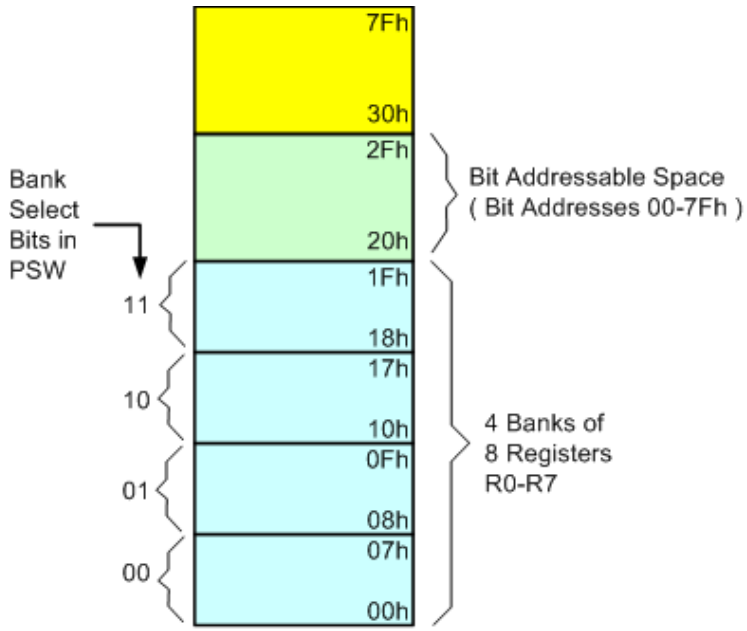
Alamat memori data internal selalu mempunyai lebar data satu byte. Pengalamatan langsung diatas 7Fh akan mengakses satu alamat memori, dan pengalamatan tak langsung diatas 7Fh akan mengakses satu alamat yang berbeda. Demikianlah pada gambar 1.4 menunjukkan 128 byte bagian atas dan ruang SFR menempati blok alamat yang sama, yaitu 80h sampai dengan FFh, yang sebenarnya mereka terpisah secara fisik

128 byte RAM bagian bawah dikelompokkan lagi menjadi beberapa blok, seperti yang ditunjukkan pada gambar 7. 32 byte RAM paling bawah, dikelompokkan menjadi 4 bank yang masing-masing terdiri dari 8 register. Instruksi program untuk memanggil register-register ini dinamai sebagai R0

sampai dengan R7. Dua bit pada *Program Status Word* (PSW) dapat memilih register bank mana yang akan digunakan. Penggunaan register R0 sampai dengan R7 ini akan membuat pemrograman lebih efisien dan singkat, bila dibandingkan pengalamatan secara langsung.

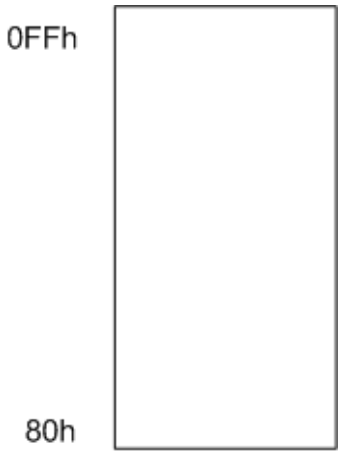


Gambar 6. Memori data internal



Gambar 7. RAM internal 128 byte paling bawah

Semua pada lokasi RAM 128 byte paling bawah dapat diakses baik dengan menggunakan pengalamatan langsung dan tak langsung. 128 byte paling atas hanya dapat diakses dengan cara tak langsung.



Gambar 8. RAM internal 128 byte paling atas

3. Special Function Register

Sebuah peta memori yang disebut ruang *special function register* (SFR) ditunjukkan pada gambar berikut. Perhatikan bahwa tidak semua alamat-alamat tersebut ditempati, dan alamat-alamat yang tak ditempati tidak diperkenankan untuk diimplementasikan. Akses baca untuk alamat ini akan menghasilkan data random, dan akses tulis akan menghasilkan efek yang tak jelas.

Accumulator

ACC adalah register akumulator. Mnemonik untuk instruksi spesifik akumulator ini secara sederhana dapat disingkat sebagai A.

Register B

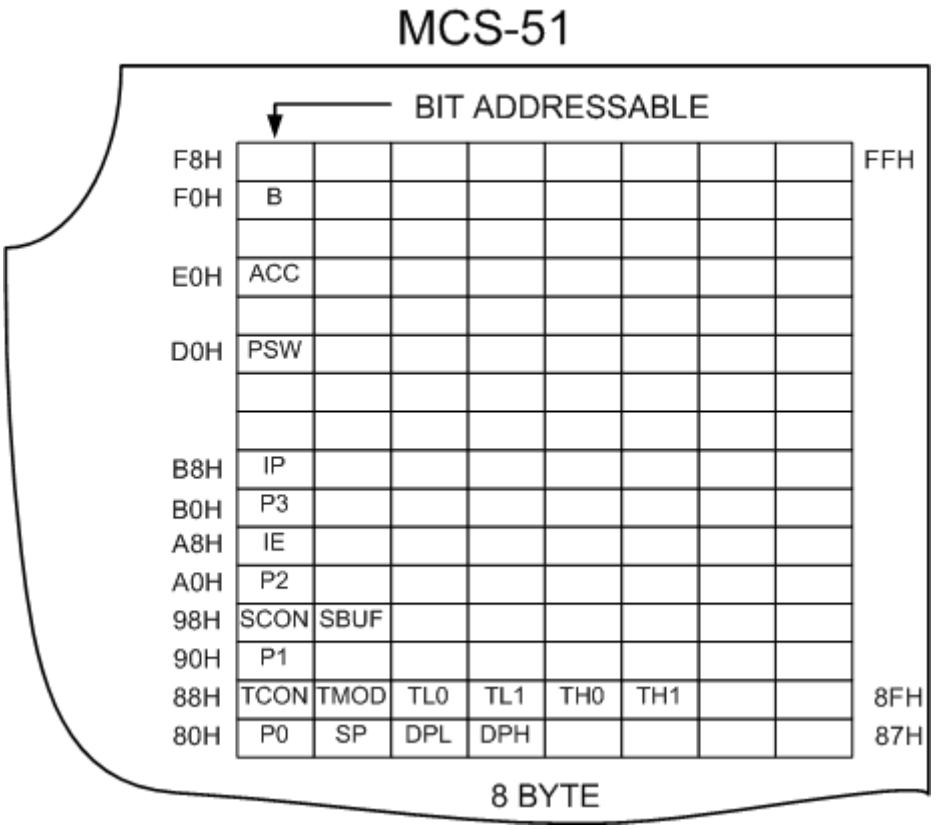
Register B digunakan pada saat opersi perkalian dan pembagian. Selain untuk keperluan tersebut diatas, register ini dapat digunakan untuk register bebas.

Stack Pointer

Register Pointer stack mempunyai lebar data 8 bit. Register ini akan bertambah sebelum data disimpan selama eksekusi *push* dan *call*. Sementara stack dapat berada disembarang tempat RAM. Pointer stack diawali di alamat 07h setelah reset. Hal ini menyebabkan stack untuk memulai pada lokasi 08h.

Data Pointer

Pointer Data (DPTR) terdiri dari byte atas (DPH) dan byte bawah (DPL). Fungsi ini ditujukan untuk menyimpan data 16 bit. Dapat dimanipulasi sebagai register 16 bit atau dua 8 bit register yang berdiri sendiri.



Gambar 9. Peta SFR

Ports 0, 1, 2 dan 3

P0, P1, P2 dan P3 adalah SFR yang ditempati oleh Port 0, 1, 2 dan 3. Menulis suatu logika 1 terhadap sebuah bit dari sebuah port SFR (P0, P1, P2 atau P3) menyebabkan pin output port yang bersesesuaian akan berada dalam kondisi logika high ‘1’, dan sebaliknya

Buffer Data Serial

Buffer serial sesungguhnya merupakan dua buah register yang terpisah, *buffer* pemancar dan *buffer* penerima. Ketika data diisikan ke SBUF, maka akan menuju ke *buffer* pemancar dan ditahan untuk proses transmisi. Ketika data diambil dari SBUF, maka akan berasal dari *buffer* penerima.

Registers Timer

Pasangan register (TH0, TL0) dan (TH1, TL1) adalah register pencacah 16 bit untuk Timer/ Counter 0 dan 1, masing-masing.

Register Control

Registers IP, IE, TMOD, TCON, SCON, dan PCON terdiri dari bit control dan status.

Program Status Word

PSW atau Program Status Word berisi bit-bit status yang berkaitan dengan kondisi atau keadaan CPU mikrokontroler pada saat tersebut. PSW berada dalam lokasi ruang SFR. Pada PSW ini kita dapat memantau beberapa status yang meliputi: *carry bit*, *auxiliary carry* (untuk operasi BCD), dua bit pemilih bank register, *flag overflow*, sebuah bit paritas dan dua flag status yang bisa didifinisikan sendiri. Bit carry dapat juga anda guakan pada keperluan operasi aritmatika, juga bisa digunakan sebagai universal akumulator untuk beberapa operasi boolean.

Tabel 2 Program Status Word

MSB							LSB
CY	AC	F0	RS1	RS0	OV	-	P

Keterangan

BIT	SYMBOL	FUNCTION
PSW.7	CY	Carry flag.
PSW.6	AC	Auxilliary Carry flag. (For BCD operations.)
PSW.5	F0	Flag 0. (Available to the user for general purposes.)
PSW.4	RS1	Register bank select control bit 1. Set/cleared by software to determine working register bank.
PSW.3	RS0	Register bank select control bit 0. Set/cleared by software todetermine working register bank.
PSW.2	OV	Overflow flag.
PSW.1	-	User-definable flag.
PSW.0	P	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of “one” bits in the Accumulator, i.e., even parity.

Bit RS0 dan RS1 dapat digunakan untuk memilih satu dari empat bank register sebagaimana ditunjukkan pada Tabel 2. Bit paritas dapat digunakan untuk mengetahuai jumlah logika '1' pada akumulator: P=1 bila pada akumulator mempunyai logika '1' yang jumlahnya ganjil, dan P=0 jika akumulator

mempunyai logika '1' yang jumlahnya genap. Dua bit yang lain PSW1 dan PSW5 dapat digunakan untuk berbagai macam tujuan

E. INTERUPSI

8051 mempunyai 5 buah sumber interupsi. Dua buah interupsi eksternal, dua buah interupsi timer dan sebuah interupsi port serial. Meskipun memerlukan pengertian yang lebih mendalam, pengetahuan mengenai interupsi sangat membantu mengatasi masalah pemrograman mikroprosesor/mikrokontroler dalam hal menangani banyak peralatan input/output. Pengetahuan mengenai interupsi tidak cukup hanya dibahas secara teori saja, diperlukan contoh program yang konkrit untuk memahami.

Saat kaki RESET pada IC mikroprosesor/mikrokontroler menerima sinyal reset (pada MCS51 sinyal tersebut berupa sinyal ‘1’ sesaat, pada prosesor lain umumnya merupakan sinyal ‘0’ sesaat), Program Counter diisi dengan sebuah nilai. Nilai tersebut dinamakan sebagai vektor reset (reset vector), merupakan nomor awal memori-program yang menampung program yang harus dijalankan.

Pembahasan di atas memberi gambaran bahwa proses reset merupakan peristiwa perangkat keras (sinyal reset diumpankan ke kaki Reset) yang dipakai untuk mengatur kerja dari perangkat lunak, yakni menentukan aliran program prosesor (mengisi Program Counter dengan vektor reset). Program yang dijalankan dengan cara reset, merupakan program utama bagi prosesor.

Peristiwa perangkat keras yang dipakai untuk mengatur kerja dari perangkat lunak, tidak hanya terjadi pada proses reset, tapi terjadi pula dalam proses interupsi. Dalam proses interupsi, terjadinya sesuatu pada perangkat keras tertentu dicatat dalam flip-flop khusus, flip-flop tersebut sering disebut sebagai ‘petanda’ (flag), catatan dalam petanda tersebut diatur sedemikian rupa sehingga bisa merupakan sinyal permintaan interupsi pada prosesor. Jika permintaan interupsi ini dilayani prosesor, Program Counter akan diisi dengan sebuah nilai. Nilai tersebut dinamakan sebagai vektor interupsi (interrupt vector), yang merupakan nomor awal memori-program yang menampung program yang dipakai untuk melayani permintaan interupsi tersebut.

Program yang dijalankan dengan cara interupsi, dinamakan sebagai program layanan interupsi (ISR - *Interrupt Service Routine*). Saat prosesor menjalankan ISR, pekerjaan yang sedang dikerjakan pada program utama sementara ditinggalkan, selesai menjalankan ISR prosesor kembali menjalankan program utama, seperti yang digambarkan dalam Gambar 10.

Gambar 10. Bagan kerja prosesor melayani interupsi

Sebuah prosesor bisa mempunyai beberapa perangkat keras yang merupakan sumber sinyal permintaan interupsi, masing-masing sumber interupsi dilayani dengan ISR berlainan, dengan demikian prosesor mempunyai beberapa vektor interupsi untuk memilih ISR mana yang dipakai melayani permintaan interupsi dari berbagai sumber. Kadang kala sebuah vektor interupsi dipakai oleh lebih dari satu sumber interupsi yang sejenis, dalam hal semacam ini ISR bersangkutan harus menentukan sendiri sumber interupsi mana yang harus dilayani saat itu. Jika pada saat yang sama terjadi lebih dari satu permintaan interupsi, prosesor akan melayani permintaan interupsi tersebut menurut perioritas yang sudah ditentukan, selesai melayani permintaan interupsi perioritas yang lebih tinggi, prosesor melayani permintaan interupsi berikutnya, baru setelah itu kembali mengerjakan program utama.

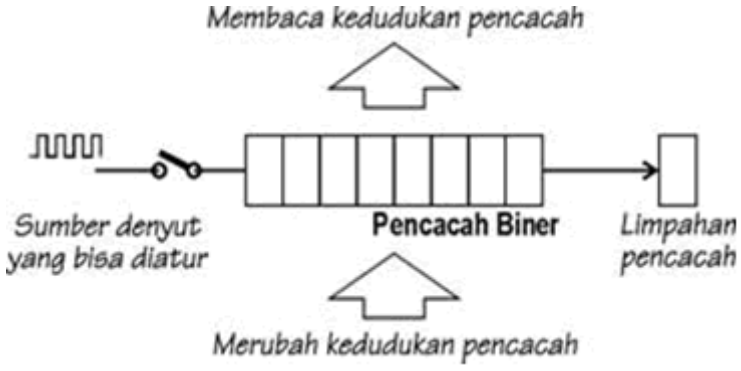
Saat prosesor sedang mengerjakan ISR, bisa jadi terjadi permintaan interupsi lain, jika permintaan interupsi yang datang belakangan ini mempunyai perioritas lebih tinggi, ISR yang sedang dikerjakan ditinggal dulu, prosesor melayani permintaan yang perioritas lebih tinggi, selesai melayani interupsi perioritas tinggi prosesor meneruskan ISR semula, baru setelah itu kembali mengerjakan

program utama. Hal ini dikatakan sebagai interupsi bertingkat (nested interrupt), tapi tidak semua prosesor mempunyai kemampuan melayani interupsi secara ini.

F. TIMER DAN COUNTER

Timer dan Counter merupakan sarana input yang kurang dapat perhatian pemakai mikrokontroler, dengan sarana input ini mikrokontroler dengan mudah bisa dipakai untuk mengukur lebar pulsa, membangkitkan pulsa dengan lebar yang pasti, dipakai dalam pengendalian tegangan secara PWM (*Pulse Width Modulation*) dan sangat diperlukan untuk aplikasi remote control dengan infra merah.Pada dasarnya sarana input yang satu ini merupakan seperangkat pencacah biner (*binary counter*) yang terhubung langsung ke saluran-data mikrokontroler, sehingga mikrokontroler bisa membaca kedudukan pancacah, bila diperlukan mikrokontroler dapat pula merubah kedudukan pencacah tersebut. Seperti layaknya pencacah biner, bilamana sinyal denyut (clock) yang diumpankan sudah melebihi kapasitas pencacah, maka pada bagian akhir untaian pencacah akan timbul sinyal limpahan, sinyal ini merupakan suatu hal yang penting sekali dalam pemakaian pencacah. Terjadinya limpahan pencacah ini dicatat dalam sebuah flip-flop tersendiri.Di samping itu, sinyal denyut yang diumpankan ke pencacah harus pula bisa dikendalikan dengan mudah. Hal-hal yang dibicarakan di atas diringkas dalam Gambar 11.

MCS-51 mempunyai dua buah register timer/ counter 16 bit, yaitu Timer 0 dan Timer 1. Keduanya dapat dikonfigurasikan untuk beroperasi sebagai timer atau counter, seperti yang terlihat pada gambar di bawah.



Gambar 11. Konsep dasar Timer/Counter sebagai sarana input

Sinyal denyut yang diumpankan ke pencacah bisa dibedakan menjadi 2 macam, yang pertama yalah sinyal denyut dengan frekuensi tetap yang sudah diketahui besarnya dan yang kedua adalah sinyal denyut dengan frekuensi tidak tetap. Jika sebuah pencacah bekerja dengan frekuensi tetap yang sudah diketahui besarnya, dikatakan pencacah tersebut bekerja sebagai timer, karena kedudukan pencacah tersebut setara dengan waktu yang bisa ditentukan dengan pasti. Jika sebuah pencacah bekerja dengan frekuensi yang tidak tetap, dikatakan pencacah tersebut bekerja sebagai counter, kedudukan pencacah tersebut hanyalah menyatakan banyaknya pulsa yang sudah diterima pencacah. Untaian pencacah biner yang dipakai, bisa merupakan pencacah biner menaik (*count up binary counter*) atau pencacah biner menurun (*count down binary counter*).

III. BAHASA ASEMBLY

Secara fisik, kerja dari sebuah mikrokontroler dapat dijelaskan sebagai siklus pembacaan instruksi yang tersimpan di dalam memori. Mikrokontroler menentukan alamat dari memori program yang akan dibaca, dan melakukan proses baca data di memori. Data yang dibaca diinterprestasikan sebagai instruksi. Alamat instruksi disimpan oleh mikrokontroler di register, yang dikenal sebagai program counter. Instruksi ini misalnya program aritmatika yang melibatkan 2 register. Sarana yang ada dalam program assembly sangat minim, tidak seperti dalam bahasa pemrograman tingkat atas (*high level language programming*) semuanya sudah siap pakai. Penulis program assembly harus menentukan segalanya, menentukan letak program yang ditulisnya dalam memori-program, membuat data konstan dan tablel konstan dalam memori-program, membuat variabel yang dipakai kerja dalam memori-data dan lain sebagainya.

1. Program sumber assembly

Program-sumber assembly (*assembly source program*) merupakan kumpulan dari baris-baris perintah yang ditulis dengan program penyunting-teks (text editor) sederhana, misalnya program EDIT.COM dalam DOS, atau program

NOTEPAD dalam Windows atau MIDE-51. Kumpulan baris-printah tersebut biasanya disimpan ke dalam file dengan nama ekstensi *.ASM dan lain sebagainya, tergantung pada program Assembler yang akan dipakai untuk mengolah program-sumber assembly tersebut.

Setiap baris-perintah merupakan sebuah perintah yang utuh, artinya sebuah perintah tidak mungkin dipecah menjadi lebih dari satu baris. Satu baris perintah bisa terdiri atas 4 bagian, bagian pertama dikenali sebagai label atau sering juga disebut sebagai symbol, bagian kedua dikenali sebagai kode operasi, bagian ketiga adalah operand dan bagian terakhir adalah komentar.

Antara bagian-bagian tersebut dipisahkan dengan sebuah spasi atau tabulator.

Bagian label

Label dipakai untuk memberi nama pada sebuah baris-perintah, agar bisa mudah menyebitnya dalam penulisan program. Label bisa ditulis apa saja asalkan diawali dengan huruf, biasa panjangnya tidak lebih dari 16 huruf. Huruf-huruf berikutnya boleh merupakan angka atau tanda titik dan tanda garis bawah. Kalau sebuah baris-perintah tidak memiliki bagian label, maka bagian ini boleh tidak ditulis namun spasi atau tabulator sebagai pemisah antara label dan bagian berikutnya mutlak tetap harus ditulis. Dalam sebuah program sumber bisa terdapat banyak sekali label, tapi tidak boleh ada label yang kembar.

Sering sebuah baris-perintah hanya terdiri dari bagian label saja, baris demikian itu memang tidak bisa dikatakan sebagai baris-perintah yang sesungguhnya, tapi hanya sekedar memberi nama pada baris bersangkutan. Bagian label sering disebut juga sebagai bagian symbol, hal ini terjadi kalau label tersebut tidak dipakai untuk menandai bagian program, melainkan dipakai untuk menandai bagian data.

Bagian kode operasi

Kode operasi (*operation code* atau sering disingkat sebagai OpCode) merupakan bagian perintah yang harus dikerjakan. Dalam hal ini dikenal dua macam kode operasi, yang pertama adalah kode-operasi untuk mengatur kerja mikrokontroler. Jenis kedua dipakai untuk mengatur kerja program assembler, sering dinamakan sebagai *assembler directive*.

Kode-operasi ditulis dalam bentuk mnemonic, yakni bentuk singkatan-singkatan yang relatip mudah diingat, misalnya adalah MOV, ACALL, RET dan lain sebagainya. Kode-operasi ini ditentukan oleh pabrik pembuat mikrokontroler, dengan demikian setiap prosesor mempunyai kode-operasi yang berlainan.

Kode-operasi berbentuk mnemonic tidak dikenal mikrokontroler, agar program yang ditulis dengan kode mnemonic bisa dipakai untuk mengendalikan prosesor, program semacam itu diterjemahkan menjadi program yang dibentuk dari kode-operasi kode-biner, yang dikenali oleh mikrokontroler. Tugas penerjemahan tersebut dilakukan oleh program yang dinamakan sebagai Program Assembler.

Di luar kode-operasi yang ditentukan pabrik pembuat mikrokontroler, ada pula kode-operasi untuk mengatur kerja dari program assembler, misalnya dipakai untuk menentukan letak program dalam memori (ORG), dipakai untuk membentuk variabel (DS), membentuk tabel dan data konstan (DB, DW) dan lain sebagainya.

Bagian operand

Operand merupakan pelengkap bagian kode operasi, namun tidak semua kode operasi memerlukan operand, dengan demikian bisa terjadi sebuah baris perintah hanya terdiri dari kode operasi tanpa operand. Sebaliknya ada pula kode operasi yang perlu lebih dari satu operand, dalam hal ini antara operand satu dengan yang lain dipisahkan dengan tanda koma.

Bentuk operand sangat bervariasi, bisa berupa kode-kode yang dipakai untuk menyatakan Register dalam prosesor, bisa berupa nomor-memori (alamat memori) yang dinyatakan dengan bilangan atau pun nama label, bisa berupa data yang siap di-operasi-kan. Semuanya disesuaikan dengan keperluan dari kode-operasi.

Untuk membedakan operand yang berupa nomor-memori atau operand yang berupa data yang siap di-operasi-kan, dipakai tanda-tanda khusus atau cara penulisan yang berlainan. Di samping itu operand bisa berupa persamaan matematis sederhana atau persamaan Boolean, dalam hal semacam ini program Assembler akan menghitung nilai dari persamaan-persamaan dalam operand, selanjutnya merubah hasil perhitungan tersebut ke kode biner yang

dimengerti oleh prosesor. Jadi perhitungan di dalam operand dilakukan oleh program assembler bukan oleh prosesor!

Bagian komentar

Bagian komentar merupakan catatan-catatan penulis program, bagian ini meskipun tidak mutlak diperlukan tapi sangat membantu masalah dokumentasi. Membaca komentar-komentar pada setiap baris-perintah, dengan mudah bisa dimengerti maksud tujuan baris bersangkutan, hal ini sangat membantu orang lain yang membaca program.

Pemisah bagian komentar dengan bagian sebelumnya adalah tanda spasi atau tabulator, meskipun demikian huruf pertama dari komentar sering-sering berupa tanda titik-koma, merupakan tanda pemisah khusus untuk komentar. Untuk keperluan dokumentasi yang intensip, sering-sering sebuah baris yang merupakan komentar saja, dalam hal ini huruf pertama dari baris bersangkutan adalah tanda titik-koma.

AT89S51 memiliki sekumpulan instruksi yang sangat lengkap. Instruksi MOV untuk byte dikelompokkan sesuai dengan mode pengalamatan (*addressing modes*). Mode pengalamatan menjelaskan bagaimana operand dioperasikan. Berikut penjelasan dari berbagai mode pengalamatan. Bentuk program assembly yang umum ialah sebagai berikut :

Label/Simbol	Opcode	Operand	Komentar
	Org	0H	
Start:	Mov	A, #11111110b	; Isi Akumulator
	Mov	R0, #7	; Isi R0 dengan 7
Kiri:	Mov	P0, A	; Copy A ke P0
	Call	Delay	; Panggil Delay
	RL	A	
	DEC	R0	
	CJNE	R0, #0, Kiri	
	Sjmp	Start	
Delay:	mov	R1, #255	
Del1:	mov	R2, #255	
Del2:	djnz	R2, del2	
	djnz	R1, del1	
	ret		

```
        end
```

Isi memori ialah bilangan heksadesimal yang dikenal oleh mikrokontroler kita, yang merupakan representasi dari bahasa assembly yang telah kita buat. Mnemonic atau opcode ialah kode yang akan melakukan aksi terhadap operand. *Operand* ialah data yang diproses oleh opcode. Sebuah opcode bisa membutuhkan 1, 2 atau lebih operand, kadang juga tidak perlu operand. Sedangkan komentar dapat kita berikan dengan menggunakan tanda titik koma (;). Berikut contoh jumlah operand yang berbeda beda dalam suatu assembly.

```
CJNE R5,#22H, aksi ; dibutuhkan 3 buah operand
MOVX @DPTR, A      ; dibutuhkan 2 buah operand
RL A               ; 1 buah operand
NOP                ; tidak memerlukan operand
```

Program yang telah selesai kita buat dapat disimpan dengan ekstension .asm. Lalu kita dapat membuat program objek dengan ekstension HEX dengan menggunakan compiler MIDE-51.

Assembly Listing

Program-sumber assembly di atas, setelah selesai ditulis diserahkan ke program Assembler untuk diterjemahkan. Setiap prosesor mempunyai program assembler tersendiri, bahkan satu macam prosesor bisa memiliki beberapa macam program Assembler buatan pabrik perangkat lunak yang berlainan. Hasil utama pengolahan program Assembler adalah program-obyek. Program-obyek ini bisa berupa sebuah file tersendiri, berisikan kode-kode yang siap dikirimkan ke memori-program mikrokontroler, tapi ada juga program-obyek yang disisipkan pada program-sumber assembly.