BAB

11

Organisasi Mikrokomputer

11.1. REVOLUSI MIKROKOMPUTER

Prinsip *organisasi* mikrokomputer adalah sama dengan komputer besar yang lain. Mungkin anda bertanya, mengapa bab ini membicarakan organisasi mikrokomputer? Untuk suatu hal, kita percaya bahwa suatu buku tentang organisasi komputer telah menjamin bab tentang organisasi mikrokomputer disebabkan pengaruh kuat teknologi mikro elektronik. Kedatangan teknologi "large-scale integration (LSI)" dan "very large scale integration (VLSI" menghasilkan berbagai macam piranti komputer yang mempunyai hubungan, pada chip integrated circuit (IC), pada pembuangan setiap perancangan sistem. Sekarang, kita dapat menggunakan pembuatan block tersebut untuk membentuk sistem digital yang lebih komplek, komputer, atau sistem komputer yang terpancang.

Kita juga dapat menggunakan bab ini sebagai acuan berbagai konsep yang diperkenalkan dalam bab-bab sebelumnya. Kita sekaran akan menyusun konsep-

konsep ini dan menggabungkannya dalam diskusi tentang organisasi mikrokomputer.

Revolusi mikrokomputer dimulai dengan "**mikroprosesor**", yaitu prosesor dengan ship-tunggal. Prosesor ini memerlukan tiga dekade dari pengenalan komputer elektronik pertama kali sebelum munculnya mikroprosesor, tetapi menguntungkan banyak dari pengalaman yang diperoleh dalam rancangan komputer yang besar. Beberapa keunggulan feature secara organisasional adalah bentuk bagian utama dari hampir semua mikroprosesor sekarang ini yang jarang dihubungkan dalam komputer yang lebih besar dan lebih mahal dibandingkan dengan beberapa tahun yang lalu. Mari kita perhatikan secara singkat tentang evolusi mikroprosesor dalam dua dekade terakhir ini.

11.2. PENGEMBANGAN SECARA TEKNOLOGI

Evolusi dari mikroprosesor dapat dibagi menjadi empat generasi. Yaitu "*first-generation microprocessor*" (mikroprosesor generasi pertama), yang diperkenalkan pada awal tahun 1970-an, yang berisi hampir mencapai 4-bit kata (yaitu, Intel 4040) atau 8-bit kata (yaitu, Intel 8080). Mikroprosesor ini dibuat denganmenggunakan teknologi PMOS dan digunakan dalam aplikasi industri sederhana dan peralatan konsumer yang sederhana pula.

"*Second-generation microprocessor*" (Mikroprosesor generasi kedua) yang muncul pada awal tahun 1973 dan berisi 8-bit kata (yaitu, Intel 8085, Zilog Z80, dan Motorola 6809). Mikroprosesor ini juga dibuat dengan teknologi PMOS, tetapi chipnya lebih besar dan kerapatannya lebih tinggi (lebih dari transistor per chip). Mikroprosesor generasi kedua mampu dalam pengalamatan spasi memori yang lebih besar (hingga 64 kilobyte) dan berisi lebih dari bagian I/O dibandingkan dengan mikroprosesor generasi pertama. Ini juga diberi karakteristik dengan kecepatan operasi yang lebih cepat, mengurangi kecepatan waktu dari power product, dan lebih handal dalam instruksinya. Aplikasi khusus dari mikroprosesor generasi kedua mencakup terminal inteligen, sistem penerimaan data, pengontrol industri yang komplek, dan sistem komunikasi.

Pada awal tahun 1978 ditandai dengan dimulainya "*third microprocessor generation*" (Mikroprosesor generasi ketiga) dengan memperkenalkan Intel 8086 dengan chip tunggal, 16 bit mikroprosesor. Pembuatan lain yang diikuti yang sesuai dengan mikroprosesor yang sama, seperti Zilog Z8000 dan Motorola 68000. Teknologi yang dominan yang digunakan untuk pembuatan mikro-

prosesor ini diberi karakteristik dengan peningkatan density chip, kemampuan pemrosesan yang lebih kuat, dan kecepatan yang lebih tinggi. Mikroprosesor generasi ketiga dapat mengalamatkan ruang memori yang besar (hingga 16 megabytes), termasuk feature virtual memori, dan memiliki kemampuan mengatasi interupsi yang lebih handal.

Intel APX-432, dengan sistem pemrosesan 32-bit, diperkenalkan dalam tahun 1981, inilah tanda adanya "*fourth generation*" (generasi keempat). Versi yang paling akhir dari Motorola 68000 juga mencakup kemampuan meng-handle kata 32-bit, seperti yang dilakukan oleh National Semicolor NS 16032 dan Texas Instrument TI 99000. Dibuat dengan teknologi HMOS dan diberi karakteristik dengan peningkatan lebih lanjut dalam kerapatan chip dan kecepatan pemrosesan, mikroprosesor generasi keempat bersaing ketat dengan mainframes. Kemampuan pengalamatan virtual memori dari spasi hingga 2^{40} byte, floating-point perangkat keras, pemisahan sistem dan perangkat lunak pemakai, dan dukungan yang lebih efisien dari pembuatan multiprosesor, ada, kecuali beberapa feature dari mikroprosesor ini. APX-432 merupakan mikroprosesor pertama yang digunakan dalam struktur bus baru yang memperkerjakan packet switching. (Packet switching akan didiskusikan dalam Bagian 12.6).

Sehubungan dengan pengembangan mikroprosesor chip adalah pengembangan sejumlah besar variasi dari "IC *support chips*", dual-port RAM, floating-point ROM, pengontrol peripheral intelligent, pengontrol terminal cluster, unit interface parallel dan serial, pengontrol direct memory access (MA), unit manajemen memori, pengontrol komunikasi multi protokol, interface komunikasi yang dapat diprogram, pengontrol I/O, dan prosesor I/O, untuk menyebutkan pandangannya.

Meskipun mikroprosesor 8-bit masih berguna untuk beberapa aplikasi, tetapi, tentu saja, relatif terbatas hingga 16 dan 32 bit mikroprosesor, khususnya ketika operasi yang besar dan komplek digunakan. Mikroprosesor 80bit juga terbatas dalam kemampuan pengalamatan memori. Mikroprosesor 8-bit secara khusus, mempunyai 16-bit alamat bus, sehingga dapat mengalamatkan hingga 65,536 (64K) bit dari memori eksternal. Sebagai tambahan, sejumlah besar operasi harus diprogram (yaitu, diimplementasikan dengan suroutine), yang menghasilkan dalam waktu pemrosesan yang lebih besar dan menggunakan ruang memori yang mempunyai nilai.

Sebaliknya bahwa mikroprosesor 16 dan 32 bit adalah lebih besar dan lebih cepat. Mereka menggunakan perangkat keras yang lebih, mampu melalui teknologi VLSI, untuk mengimplementasikan variasi fungsi yang akan diimplementasikan dalam perangkat lunak. Jika mereka dapat memproses kata yang lebih

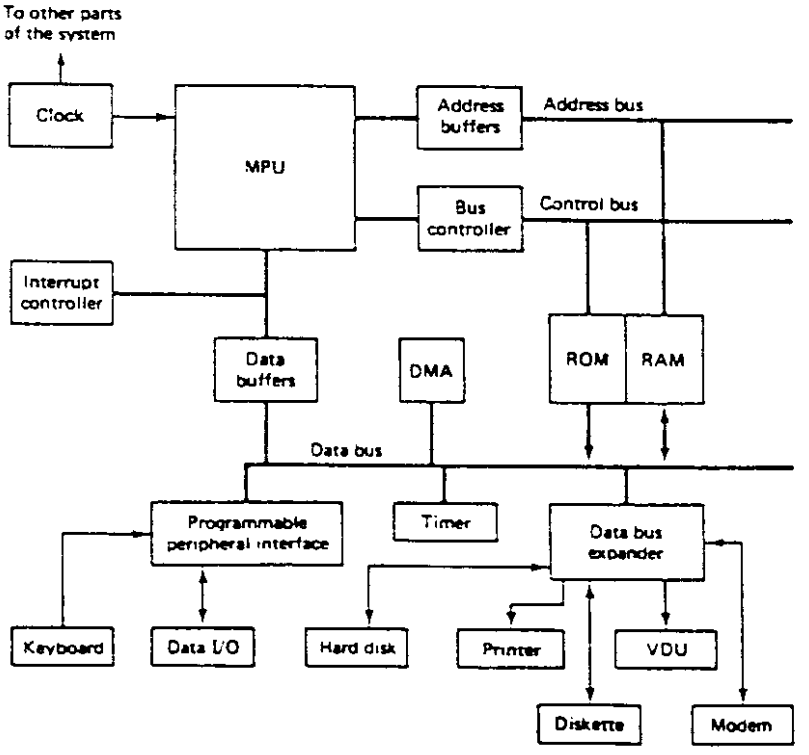
panjang, maka mikroprosesor ini telah meningkat kemampuan pengalamatan memori termasuk pengalamatan mode, yang memungkinkannya untuk mengirimkan berjuta kata secara langsung. Mikroprosesor berisi register yang banyak, yang mendukung serangkaian instruksi yang lebih besar, dan dapat mengerjakan beberapa yang lebih cepat dan lebih efisien. Frekuensi clock dalam mikroprosesor ini secara khusus berada antara 8 hingga 20 Mhz, berlawanan dengan mikroprosesor 8-bit dimana frekuensi clocknya hingga 6 MHz. Suatu arsitektur baru yang digabungkan ke dalam mikroprosesor 16 dan 32-bit akan mempercepat peningkatan beberapa pesan dari magnitude, akhirnya mikroprosesor yang baru, secara khusus menawarkan kemampuan manajemen memori yang memungkinkannya digunakan dalam lingkungan multitasking, multiuser.

11.3. PANDANGAN UMUM DARI SISTEM MIKROKOMPUTER

Gambar 11-1 menunjukkan suatu diagram blok dari sistem mikrokomputer. Unit pemrosesan pusat adalah chip mikroprosesor, yang disusun dengan MPU (*microprosesor unit*), yang dihubungkan dengan memori dan unit peripheral melalui data, alamat, dan kontrol bus. Berbagai variasi chip pendukung juga ditunjukkan, termasuk generator jam, pengontrol bus, pengontro interupsi, dan pengontrol direct memory access (DMA). Semua komponen tersebut dapat dikelompokkan ke dalam tiga unit fungsional yang berbeda,: yaitu "**microprocessing unit (MPU)**", "**memory**", dan "**input/output (I/O)**". (Tentu saja, MPU tak lain dari CPU). Komunikasi antarunit tersebut dikontrol dengan MPU.

MPU akan membuat program yang disimpan dalam memori. Program ini akan memanipulasi data dan membuat keputusan berdasarkan pada data. Program ini menyediakan waktu dan signal kontrol untuk pengontrol bus, I/O, dan memori. ROM dan RAM akan menyimpan program dan data. RAM menyediakan instruksi dan data untuk MPU atas permintaan dan menerimaan data baru dari MPU untuk disimpan. ROM akan menyimpan hanya sistem program. Unit I/O menyediakan hubungan komunikasi antara MPU dan dunia luar dan membiarkan MPU meng-input data ke dan meng-output data dari bagian eksternal peripheral seperti keyboard, printer, monitor, dan disk atau unit penyimpanan disket.

MPU akan menggunakan **buses** (jalur komunikasi) untuk berkomunikasi dengan berbagai bagian dari sistem mikrokomputer. Kita membedakan tiga type dari buses, yaitu: **data bus**, alamat bus, dan kontrol bus. **Data bus** akan mengi-



Gambar 11-1 Diagram blok sederhana dari sistem mikrokomputer.

rimkan data ke dan dari MPU. Data dalam hal ini berisi angka yang disimpan dalam memori yang harus dibaca oleh MPU atau angka dalam register MPU yang harus ditulis ke dalam memori. Data bus, oleh karena itu, disebut dengan "*bidirectional*", karena data dapat berjalan pada semua tujuan, dari memori ke MPU atau dari MPU ke memori. Untuk memberikan pengaruh pada transfer data ini, MPU akan menggunakan baris kontrol "*read/write*". Misalnya, jika baris kontrol ini diatur dengan angka 1, maka MPU akan membaca data dari data bus, tetapi jika diatur kembali dengan angka 0, maka MPU akan menempatkan (menulis) data pada data bus. **Address bus** memungkinkan MPU untuk memilih lokasi memori atau tujuan (bagian) I/O untuk transfer data. Address bus dapat berupa "*undirectional*" karena informasi yang berjalan hanya dalam satu tujuan, yaitu dari MPU ke memori atau I/O.

Pada beberapa rancangan mikroprosesor, alamat dan data bus adalah (minimal secara bagian) di-multiplexed (misalnya, Intel 8085 dan 8086). Dengan kata lain, bus yang sama dapat digunakan untuk transfer data bidirectional atau pemilihan alamat undirectional. Rancangan mikroprosesor lain, seperti Motorola 68000, menggunakan data terpisah dan alamat bus.

Control bus berisi berbagai variasi baris kontrol yang diperlukan untuk mengontrol operasi MPU (misalnya, baris kontrol *read/write*). Bus biasanya merupakan bidirectional; namun demikian beberapa mikroprosesor menggunakan kontrol bus unidirectional. Dalam kasus selanjutnya, beberapa baris kontrol, hanya berjalan pada MPU ketika yang lainnya berjalan hanya ke dalam MPU.

Program yang mengontrol operasi sistem mikrokomputer terdiri atas instruksi-instruksi yang dapat dibuat oleh mikroprosesor. Serangkaian **instruction** dapat dibagi ke dalam kelompok menurut tipe operasi yang menunjukkan instruksi. Kelompok ini mencakup aritmatika, logika, transfer data, input/ouput, kontrol, dan cabang instruksi.

Beberapa organisasi dan hubungan adalah mungkin di antara blok building internal sistem mikrokomputer. Organisasi tertentu akan menambah pembuatan pada satu kelompok instruksi ketika pengurangan yang lainnya. Oleh karena itu, tidak ada sesuatu yang disebut "best" (terbaik) arsitektur mikrokomputer. Lain dengan organisasi mikrokomputer terbaik dalam aplikasi yang terikat. Misalnya, jika mikrokomputer digunakan untuk hampir semua data logging, maka organisasinya harus dioptimiskan untuk pemasukan memori dan operasi I/O. Jika ini digunakan secara pokok untuk meremukkan angka, maka diperlukan arsitektur yang diorientasikan dengan/melalui hitungan matematika.

11.4. UNIT PEMROSESAN SINGLE-CHIP (MPU)

Organisasi internal MPU, seperti CPU, terdiri dari (1) **register set**, (2) **arithmatic and logic unit (ALU)**, dan (3) **control logic unit (CLU)**. Transfer data dalam MPU diselesaikan melalui satu atau lebih bus internal. Angka bit dimana MPU (khususnya, ALU) dapat memproses secara paralel dengan menentukan **word length** pokok dari mikroprosesor. Kata MPU merupakan elemen pokok yang dapat dialamatkan dalam memori dan menentukan elemen data pokok dari mikroprosesor dan menentukan lebar data bus. Misalnya, panjang kata 16 bit mikroprosesor adalah 16 bit, memerlukan 16 bit-wide data bus.

Informasi yang ditangani oleh mikroprosesor adalah dua tipe: yaitu instruksi dan data (operand). Panjang kata pokok biasanya bebas instruksinya atau panjang operand-nya. Suatu instruksi mungkin satu, dua, atau lebih panjang kata dan dapat tidak berisi apapun (operand), dapat juga berisi operand satu, dua, atau lebih. Mikroprosesor dapat mendukung beberapa tipe data (operand), seperti aritmatika, logika, boolean, dan data alphanumerik.

Register Set

Register ser terdiri atas register-register dengan tujuan umum, dimana biasanya ada antara 8 hingga 32 (tergantung pada mikroprosesor khusus), dan register dengan tujuan tertentu, yang masing-masing digunakan untuk fungsi khusus dan digunakan langsung atau tidak langsung oleh instruksi program. **General-purpose register** (register dengan tujuan umum) dapat digunakan (oleh instruksi) sebagai akumulator, sumber atau tujuan register data, atau register alamat yang berisi pointer memori atau nilai indeks. Sebagai tambahan, register dengan tujuan umum ini dapat digunakan untuk memberikan fasilitas perpindahan block, untuk mengijinkan adanya (stacking), dan untuk menambah alamat indeks. Beberapa mikroprosesor membebani/memberikan batasan pada beberapa register dengan tujuan umum. Misalnya, beberapa register dapat digunakan hanya untuk memori alamat atau hanya sebagai register data. Meskipun batasan ini merupakan batas pemrograman yang fleksibel (jika kita menggunakan register ini untuk fungsi lain, kita pertama kali harus menyimpan isinya secara sementara), namun batasan ini memungkinkan perancang komputer untuk bekerja sama dengan format instruksi yang lebih kompak.

Sebaliknya, **Special-purpose register** (register dengan tujuan tertentu), digunakan untuk fungsi khusus. **Program counter** (PC) biasanya terdiri atas alamat memori tempat kata instruksi berikutnya akan diambil. Hampir semua mikroprosesor mencakup lebih dari satu "**accumulator** (ACC)", fungsi-fungsi dimana menyimpan satu operand agar dioperasikan oleh ALU dan menyimpan hasil dari operasi ALU. **Instruction register** (IR) berisi instruksi opcode. **Index register** menyimpan angka konstanta yang digunakan dalam penghitungan alamat memori yang efektif dalam mode pengalamatan indeks.

Hampir semua mikroprosesor bergabung dengan register tujuan tertentu, yang disebut dengan **stack register**, yang mengambil pada porsi set RAM untuk operasi subroutine atau untuk menangani kondisi perkecualian (misalnya, interupsi). Stack akan menyimpan alamat instruksi yang harus dibuat

setelah subroutine lengkap. Dengan demikian instruksi berikutnya yang diambil akan menjadi subroutine yang pertama, dan ketika mikroprosesor mengambil instruksi yang terakhir dari subroutine (yang selalu berupa instruksi *return*), mikroprosesor tersebut akan mengganti isi PC dengan alamat paling atas dari stack dan menyimpulkan pembuatan program utama. (Nested subroutine) juga dapat didukung, jika ada cukup memori untuk menyimpan alamat return yang perlu.

Hampir semua mikroprosesor 16- dan 32-bit menggunakan **segment register** untuk mengimplementasikan pemetaan memori ketika pemasukan memori utama. Mekanisme ini akan menterjemahkan alamat logika program ke dalam alamat fisik. Beberapa mikroprosesor berisi **memory refresh register** yang menyediakan secara otomatis transparan refresh dari dynamika RAM (lihat bagian 3.8). Biasanya, mikroprosesor berisi **vector interrupt register** yang memungkinkan tabel interupsi untuk menempatkan *dimana saja* dalam memori. Dalam hal ini, semua yang MPU harus mengambil instruksi pertama dari interrupt service routine merupakan alamat *tidak langsung* dari lokasi memori yang berisi instruksi tersebut.

Angka **temporary register**, yang tidak dapat dimasukkan pada pemakai, ada juga di dalam hampir semua mikroprosesor dan digunakan untuk menangani operand lanjutan atau hasil penyimpanan sementara dari operasi. Misalnya, **status register** berisi berbagai indikasi, yang disebut dengan **flag**, yang disediakan oleh mikroprosesor. Tiap flag digunakan untuk menunjukkan status dari kondisi mikroprosesor tertentu sebagai hasil dari suatu operasi. Beberapa flag digunakan langsung oleh instruksi, ketika angka-angka flag yang lainnya diuji di bawah program kontrol untuk menentukan serangkaian instruksi yang mengikutinya. Berikut ini beberapa contoh dari flag yang umum:

1. *Sign flag*. Flag ini digunakan untuk menunjukkan tanda dari hasil suatu manipulasi data atau operasi transfer data. Ini memerlukan angka *most significant bit* (signbit) dari hasil dan dapat diuji untuk sign yang positif (sign bit = 0) atau sign negatif (sign bit = 1). Flag ini menganggap angka significant bit (*msb*) dari hasil ketika angka-angka yang tidak diberi sign digunakan. Mikroprosesor tidak mempunyai cara dengan mengetahui jika data menunjukkan angka yang ber-sign atau yang tidak atau informasi nonnumerik, ini tergantung pada programer untuk menjaga track ini.
2. *Carry flag*. Flag ini diset ke 1 jika operasi aritmatika, seperti penambahan atau pengurangan, akan menghasilkan di dalam carry atau mengeluarkan *msb*, jika sebaliknya maka flag ini di-set dengan 0.

3. *Auxiliary carry flag*. Flag ini digunakan untuk memberikan pengaruh status dari carry lanjutan dalam dua-digit BCD (desimal) dari operasi aritmatika.
4. *Parity flag*. Parity flag biasanya menunjukkan odd parity dari suatu hasil. Jika angka satu ada sebagai hasilnya, maka parity flag di set dengan 1, jika sebaliknya, maka parity flag di set dengan 0 (yang membuat total angka satu odd/aneh).
5. *Interrupt flag*. Flag ini digunakan untuk memungkinkan atau tidak memungkinkan adanya interupsi. Jika interrupt flag di set dengan 0, maka dapat diinteruptsi, dan jika di set dengan 1, maka tidak dapat.

Beberapa mikroprosesor mempunyai indikator status tambahan. Misalnya, *overflow flag*, yang digunakan untuk menunjukkan overflow yang dihasilkan dari aritmatika komplemen 2 (lihat bagian 2.3). Perhatikan bahwa flag ini berbeda dengan carry flag dan dua di antaranya tidak dapat saling ditukar. Beberapa mikroprosesor berisi subtract flag yang di set dengan 1 untuk operasi termasuk pengurangan dan diset dengan 0 untuk memasukkan operasi penambahan.

Beberapa mikroprosesor 16- dan 32-bit dapat beroperasi dalam **user mode** maupun dalam **system (supervisor) mode**. Proses dua mode tersebut disediakan untuk menambah keamanan dalam sistem. Hampir semua pemakai program membuat dalam mode pemakai ketika mengoperasikan sistem yang membuat dalam mode sistem. Hampir semua instruksi membuat yang sama dalam mode tetapi beberapa instruksi, seperti instruksi I/O, dan interrupt, dapat dibuat hanya dalam mode sistem. Sehubungan dengan hal itu, untuk menunjukkan status tertentu dari prosesor, maka status register harus berisi *user/system flag*.

Akhirnya, beberapa mikroprosesor mempunyai *trap flag* yang diatur untuk membuat interupsi setelah pembuatan masing-masing instruksi. Feature ini memungkinkan programer untuk mengoperasikan mikroprosesor dalam *single mode*, yang sangat berguna dalam pembuatan program.

Aritmatika dan Logic Unit (ALU)

Aritmatic and Logic Unit (ALU) menunjukkan operasi aritmatika, logika, dan manipulasi data pada angka biner. Pada berbagai mikroprosesor, ALU mampu dalam penambahan, pengurangan, perbandingan, pengambilan, dan operasi logika yang ditentukan dengan *control logic unit* (CLU). Pada hampir semua mikroprosesor aritmatika komplemen 2 digunakan untuk menunjukkan angka negatif.

Lebar ALU berhubungan dengan panjang kata pokok dari MPU. ALU memerlukan register penyimpanan untuk menyimpan input kata (operand) dan latch (yang kadang-kadang internal pada ALU) untuk menyimpan hasil sementara dari operasi ALU. Input disimpan dalam akumulator, register sementara, atau dalam register dengan tujuan umum dalam MPU, atau berasal dari memori utama. Hasil dalam operasi ALU dikirimkan melalui data bus pada akumulator, pada register dengan tujuan umum dalam MPU, untuk menentukan lokasi memori utama, atau untuk tujuan lain. Jika beberapa operasi ALU berpengaruh pada satu flag atau lebih, maka hasil dari operasi ALU akan membawa status signal untuk men-set atau men-set kembali flag tersebut dalam register status.

Pada umumnya, operasi ALU dapat diklasifikasikan ke dalam operasi single-operand atau dua-operand. Berikut ini beberapa contoh dari **single-operand** operasi ALU yang umum:

1. INCREMENT; Menambah angka operand dengan 1.
2. DECREMENT; Mengurangi angka operand dengan 1.
3. CLEAR; Men-set operand pada 0.
4. SHIFT; mengambil operand ke sebelah kiri (dengan menggunakan instruksi SHIFT-LEFT) atau ke sebelah kanan (dengan menggunakan instruksi SHIFT-RIGT) dengan satu posisi bit atau lebih. Pada hampir semua mikroprosesor, bit yang diambil dari operand tidak hilang tetapi diambil sebagai pengganti ke dalam flag carry bit dari register status.
5. ROTATE; modifikasi dari operasi SHIFT dimana flag carry dan operand membentuk ring (sirkulasi) register pengambil (shift register). Di sini, juga memungkinkan adanya operand ROTATE-LEFT atau ROTATE-RIGHT.
6. INVERT; Meng-komplemen semua bit dari operand.

Di antara instruksi ALU **two-operand** yang paling umum adalah sebagai berikut:

1. ADD: Menghasilkan jumlah binary dari dua operand. Jika suatu operasi menghasilkan di dalam carry-out dari most significant bit, maka carry flag dalam register status diset dengan angka 1.
2. SUBTRACT: Mengurangi satu operand dari yang lain. Biasanya subtrahend diekspresikan dalam komplemen 2 dan ditambahkan untuk minuend. Carry terakhir yang dibuat dengan operasi ini disimpan dalam carry flag.
3. COMPARE: Menentukan operand terbesar di antara dua operand dengan pengurangan. Hasilnya mungkin positif, negatif atau nol akan mempengaruhi kondisi flag.

4. AND, OR, XOR: Beroperasi secara logis pada hubungan bit dari dua operand.

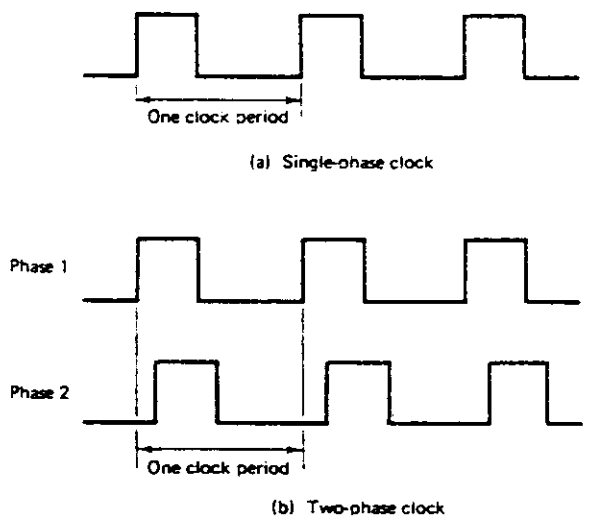
Operasi aritmatika ALU dalam mikroprosesor 8 bit dibatasi dengan penambahan dan pengurangan biner. Bahkan, mikroprosesor 6- dan 32- bit yang lebih hebat menyediakan untuk aritmatika sign dan unsign, perkalianan pembagian, aritmatika desimal, dan bahkan aritmatika floating-point.

Control Logic Unit (CLU)

Control Logic Unit (CLU) mempunyai dua tugas dari pe-sinkron-an operasi unit internal dari MPU, seperti ALU dan register, dan operasi modul mikrokomputer yang lain, seperti port I/O dan memori. Fungsi dari CLU adalah untuk mengambil instruksi dari memori dan men-decode-nya dan kemudian membuat pengaturan waktu yang tepat dan signalkontrol yang diperlukan oleh MPU untuk pembuatan instruksi tersebut. Dalam dua aturannya, CLU juga membuat signal waktu dan signal kontrol yang dikirimkan, melalui kontrol bus, ke komponen lain dari sistem mikrokompoter dan menangani dan merespon signal eksternal seperti interupsi.

Untuk menyediakan sinkronisasi yang diperlukan di antara semua elemen, sistem mikrokomputer begabung dengan **clock pulse generator**. Signal waktu yang disediakan oleh lingkaran jam adalah periodik dan **single-phase** atau **multi-phase**. Signal multiphase berisi signal periodik yang disinkronkan dengan yang lainnya tetapi biasanya di luar phase. Gambar 11-2 menunjukkan contoh single-phase dan dua pashe signal jam. Meskipun mikroprosesor 8-bit telah digunakan dalam implementasi jam dua phase, hampir semua mikroprosesor yang lebih baru (8-bit dan 32-bit mikroprosesor) sekarang menggunakan implementasi jam single-phase.

Control logic unit sebenarnya merupakan komputer dengan tujuan tertentu dalam MPU dan memerlukan suatu program untuk menuntunnya dalam pembuatan suatu instruksi. Pada hampir semua mikroprosesor, CLU merupakan **microprogrammed**. Yaitu, signal waktu dan signal kontrol diperlukan untuk mengambil dan membuat suatu isntruksi yang dibuat dengan pembuatan serangkaian **microinstruction** (atau **microcode**) yang residen dalam **control memory**, yang biasanya berupa ROM. Mikro instruksi disimpan dalam ROM yang membentuk **microprogram** yang secara lengkap menentukan signal kontrol. Dengan demikian masing-masing **instruksi mikroprosesor** yang sebaliknya dapat mengembalikan kontrol yang diperlukan dan signal waktu untuk pembuatan instruksi makro. Dengan demikian pembuatan alamat, pengambilan, dan pemberian



Gambar 11-2 Signal waktu.

decode diperoleh dengan cara yang sama dengan instruksi makro. Demikian juga, kita dapat memasukkan dalam program mikro cabang yang tidak kondisional, cabang kondisional pada status flag, dan subroutine akan memanggil dan mengembalikan. Secara khusus, berbagai operasi mikro, ditunjukkan dengan instruksi makro, termasuk seperti sumber pemilihan operand ALU, fungsi ALU, kontrol carry, shift kontrol, interrupt kontrol, dan kontrol data in/out. Seperti yang telah kita ketahui, penggunaan mikroprograming akan mengurangi pertimbangan keberadaan perangkat keras dan menyediakan pesan organisasi CLU yang tinggi dan dapat secara cepat dan mudah dimodifikasi dan didiagnosa.

Mari kita lihat secara ringkas serangkaian operasional microprogram CLU. Ketika instruksi mikro diambil dari memori, opcode-nya digunakan untuk menentukan ROM permulaan alamat dari segmen program mikro (**microroutine**) yang mengimplementasikan instruksi. Setelah menerima alamat permulaan mikroroutine, ROM akan mendukung instruksi mikro. Ini biasanya mengirimkan instruksi mikro pada pipeline register (lihat Bab-09) yang melayani sebagai buffer dan meningkatkan penampilan dan kecepatan dengan overlapping pembuatan instruksi mikro dan pengambilan instruksi mikro selanjutnya. Satu kali semua instruksi mikro dalam mikroroutine dibuat, maka pembuatan instruksi mikro akan lengkap. Jika instruksi makro selanjutnya diambil, maka prosesnya akan diulang.

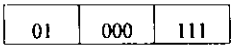
11.5 SET INSTRUKSI

Set Instruksi mencakup instruksi aritmatika dan logika, instruksi transfer data, instruksi input/output, instruksi cabang, dan instruksi kontrol. Semua mikroprosesor mendukung hampir semua tipe instruksi tersebut, tetapi serangkaian instruksi sangat bervariasi dari mikroprosesor ke mikroprosesor lainnya, seperti dalam format instruksi. Setiap pembuat mikroprosesor akan menyediakan pemakai dengan daftar berbagai variasi instruksi yang ada dan formatnya.

Instruksi-instruksi dibuat dengan (penggabungan) angka byte. Bagian dari instruksi yang menentukan apa yang disebut dengan **operation code (opcode)**. Bagian dari instruksi yang berisi informasi, data, atau alamat yang diperlukan untuk kelengkapan suatu pembuatan instruksi disebut dengan **operand**. Instruksi-instruksi mungkin bervariasi panjangnya, tetapi masing-masing mempunyai format yang sama; opcode selalu diikuti dengan satu operand atau lebih. Instruksi yang memerlukan lebih dari satu operand disebut dengan instruksi **multi-opernad**. Untuk memberikan lokasi dan pemasukan suatu operand, mikroprosesor akan menggunakan satu dari berbagai **addressing mode** yang ada.

Jika setiap instruksi diulang seperti pola bit, maka memungkinkan untuk memprogram mikroprosesor dengan menuliskan kode biner dari setiap instruksi. Ini disebut dengan **machine language programming**. Secara jelas, bahwa proses ini memerlukan waktu, cenderung mempunyai kesalahan, dan tidak bisa diharapkan untuk sejumlah besar instruksi. Namun demikian, biasanya suatu alternatif pendekatan pemrograman tingkat rendah, yang disebut dengan **assembly language programming**, telah digunakan. Dalam assembly language. setiap instruksi bahasa mesin ditunjukkan dengan simbol yang disebut dengan **mnemonic**, bukannya dengan pola bit. Translasi one-to-one antara instruksi ber-simbol tersebut dan hubungan kode bahasa mesin ditunjukkan dengan program yang disebut dengan **assembler**. Ada dua tipe dari assembler: yaitu **self-assembler**, yang berjalan pada mikrokomputer untuk membuat kode bahasa mesin, dan **cross-assembler**, yang berjalan pada komputer yang berbeda dengan yang pertama dimana kode mesin yang membuat akan dibuat.

Untuk mengilustrasikan perbedaan antara bahasa mesin dan instruksi bahasa assembly, perhatikan operasi transfer dari register ke register B ← (A). Suatu instruksi bahasa mesin untuk transfer semacam itu berupa



dimana pola bit 01 menunjukkan opcode, dan 000 dan 111 menunjukkan alamat tujuan register B dan alamat sumber register A, secara respektif. 'Hubungan instruksi bahasa assembler akan berupa sebagai berikut:

MOV B,A

Dengan demikian tugas assembler akan membuat kode 01 untuk instruksi MOV (pindah), 000 untuk menunjukkan register B, dan 111 menunjukkan sumber register A.

Tipe Instruksi

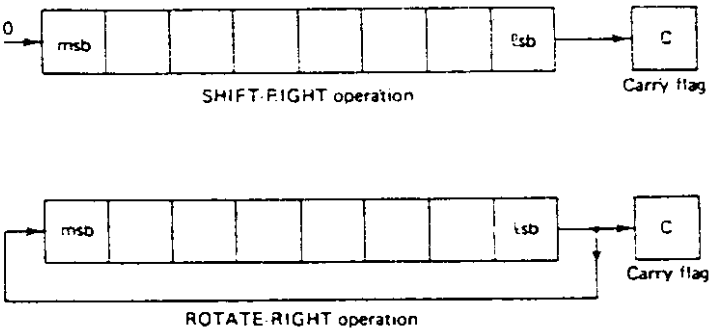
Instruksi aritmatika. Instruksi aritmatika menyediakan untuk manipulasi data aritmatika. Instruksi khusus dalam jenis ini adalah ADD, ADD WITH CARRY, COMPLEMENTS 1' dan 2 , dan MULTIPLY dan DEVIDE. Opcode pada masing-masing instruksi tersebut diikuti, biasanya, dengan alamat sumber dan tujuan register dimana operand yang berhubungan dengan instruksi yang disimpan.

Misalnya, instruksi INCREMENT akan meningkatkan dengan 1 isi sumber register dan menyimpan hasil dalam register tujuan. Demikian juga, instruksi COMPLEMENT 2 akan mengurangi isi sumber register dari 2 (dimana n merupakan panjang kata) dan menempatkan hasilnya ke dalam register tujuan.

Instruksi Logika. Jenis instruksi ini menyediakan manipulasi untuk data logika. Instruksi khusus jenis ini adalah AND, OR, XOR, (eksklusif OR), NOT, ROTATE, SHIFT, dan COMPARE. Format dari instruksi ini sama dengan instruksi aritmatika.

Misalnya, instruksi AND akan secara logika AND (bit demi bit) isi sumber tujuan register dan menyimpan hasilnya dalam register tujuan. Instruksi ROTATE-LEFT *n* BITS akan memutar isi register tujuan kepada sebelah kiri dengan bit *n* dan menyimpan hasilnya kembali dalam register tujuan. Bit terlahir yang diputar akan diperoleh dalam bit flag carry dari register status. Perbedaan antara instruksi ROTATE dan SHIFT ditunjukkan dalam Gambar 11-3. Instruksi COMPARE akan menentukan (dengan pengurangan) apakah isi register tujuan lebih besar atau sama atau kurang dari register sumber dan mengatur status flag yang semestinya untuk merefleksi hasilnya.

Instruksi transfer data. Instruksi transfer data memungkinkan transfer antara informasi antara dua register MPU atau antara lokasi memori utama dan register MPU. Instruksi ini termasuk MOVE, EXCHANGE DATA, dan LOAD.



Gambar 11-3 Operasi SHIFT dan ROTATE.

Mari kita perhatikan dua instruksi tersebut. Instruksi MOVE DATA akan menyimpan isi register sumber ke dalam register tujuan. Dalam instruksi LOAD REGISTER IMMEDIATE, data merupakan bagian dari instruksi. Dengan demikian instruksi ini dapat digunakan untuk mengenalkan konstanta dalam program.

Instruksi input/output. Jenis instuksi ini menyediakan untuk mentransfer informasi antara MPU dan port input atau output. Instruksi ini secara khusus mencakup input data dan output data dan operasi data bus selama I/O.

Ada dua tipe pokok dari operasi I/O, yatu: memory-mapped I/O, dimana bagian input/output dihubungkan sebagai lokasi memori virtual, dan I/O mapped I/O, dimana bagian input/output bebas dari memori. Dalam **memory-mapped I/O**, bagian input/output dipasang pada bus alamat. Setiap bagian input diperlakukan seperti bagian dari memori dengan mempunyai beberapa data yang ditulis ke dalamnya. Dengan kata lain, bahwa port I/O diberikan susunan alamat dan dimasukkan seolah-olah mereka itu lokasi memori. Konsekuensinya, hasil memory-mapped I/O tidak memerlukan instruksi tertentu jika seluruh instruksi memory-reference dalam serangkaian instruksi dari mikroprosesor dapat juga mereferensi bagian I/O.

Dalam operasi **I/O-mapped I/O**, tidak diperlukan referensi langsung pada alamat memori. Transfer informasi dikerjakan di bawah kontrol program dan diberi tanda dengan signal interrupt yang dibuat oleh bagian yang memerlukan layanan MPU. Jika fasilitas **direct memory access (DMA)** ada, maka bagian peripheral dapat memasukkan memori utama langsung, dengan melalui MPU.

Instruksi cabang.. Instruksi cabang menunjukkan suatu bagian penting dari serangkaian instruksi. Instruksi ini menyediakan pemakai dengan arti memberikan alternatif serangkaian yang normal dari pembuatan instruksi, kemungkinan sebagai hasil dari mekanisme pembuatan keputusan. Instruksi cabang secara khusus merupakan lompatan kondisional dan nonkondisional, instruksi routine, dan interupsi perangkat lunak.

Suatu contoh dari lompatan kondisional adalah instruksi JUMP-ON-CARRY. Jika flag carry diatur, isi register alamat data akan ditransfer ke program counter dan byte yang masih ada dari instruksi diintepretasikan sebagai alamat instruksi berikutnya yang harus diambil. Jika carry flag diatur kembali, maka program counter tidak akan berubah, dan suatu instruksi tidak mempunyai pengaruh pada perjalanan program. Dalam instruksi nonkondisional, program counter selalu berubah untuk memungkinkan adanya suatu lompatan.

Instruksi subroutine dan perangkat lunak memungkinkan pemakai mengeluarkan program utama dari titik tertentu dan kemudian memasukkan kembali pada tempat yang sama atau tempat lain yang ditentukan relatif lebih baik. Pengeluaran dan pemasukan kembali dapat dikerjakan dengan instruksi JUMP dan RETURN.

Instruksi kontrol. Instruksi ini mencakup instruksi interupsi, no operation (NOP), dan HLAT atau (WAIT). Interup hard(ware), merupakan kebalikan dari interup soft(ware), tidak disebabkan oleh instruksi program tetapi oleh bagian yang memerlukan komunikasi dengan MPU. Jika terjadi interup, maka cabang mikroprosesor akan keluar dari program dan memasukkan subroutine yang khusus ditulis untuk menangani adanya interup.

Ada dua tipe pokok dari interup; yaitu: maskable dan nonmaskable. **Maskable interrupt** memungkinkan secara sementara dengan instruksi tertentu seperti INTERRUPT DISABLE. Pemakai dapat menunjukkan bagian dari program yang tidak harus diinterupsi dengan permulaan setiap bagian dengan instruksi INTERRUPT DISABLE dan mengakhiri bagian tersebut dengan instruksi INTERRUPT ENABLE. Sebaliknya, **nonmaskable interrupt** tidak dapat menjadi disable dengan instruksi perangkat lunak. Jika terjadi interup nonmaskable, maka perjalanan program akan terinterupsi.

Jika mikrokomputer berisi lebih dari satu bagian yang memerlukan interup program, maka mikrokomputer tersebut harus menentukan bagian mana (jika ada) yang telah membuat suatu permintaan dan bekerja padanya. Mikroprosesor menyelesaikan ini dengan dua cara pokok. Yang pertama adalah skema **vectored-interrupt**, dimana sirkuit logika eksternal menyediakan mikroprosesor (atas data

bus) dengan permulaan alamat dari service subroutine yang digabungkan dengan bagian interupsi. Skema ini biiasanya diprioritaskan untuk memecahkan konflik yang muncul jika dua atau lebih bagian interupsi MPU secara simultan. Metode kedua adalah skema **polled-interrupt**, dimana semua bagian mengirimkan permintaan interupsinya pada baris kontrol tunggal, dengan demikian mengaktifitaskan bagian routine interupsi umum. Routine ini, sebaliknya, dengan sukses dapat menginterogasi permintaan interupsi status bit dari masing-masing bagian. Jika bagian interupsi ditemukan, maka routine interupsi umum akan melompat pada routine yang menangani bagian tersebut.

11.6. MODE PENGALAMATAN

Suatu mikrokomputer disebut dengan **byte-addressable** jika setiap byte dalam memori mempunyai alamat yang berbeda. Jika setiap kata mempunyai alamat yang berbeda tetapi setiap byte bukan merupakan mikrokomputer maka disebut dengan **word-addressable**. Misalnya, byte-addressable mikrokomputer dengan 16-bit alamat field mempunyai spasi alamat maksimal 2^{16} *bytes* kebalikan dari 2^{16} *kata*, jika merupakan mikrokomputer word-addressable.

Hampir semua mikroprosesor menggunakan bervariasi **addressing mode**. Angka mode pengalamatan dalam mikroprosesor menunjukkan ekstensi dari serangkaian instruksi dan secara efisien menangani berbagai variasi struktur data. Mode pengalamatan telah didiskusikan secara mendalam dalam Bab 5 dan secara singkat diterangkan sedikit di sini.

Meskipun terminologi digunakan secara berbeda dari satu pembuatan mikroprosesor dengan yang lain, daftar berikut ini menerangkan mode pengalamatan yang biasa dipakai:

1. *Direct addressing.* Alamat memori yang efektif dimana operand yang dilokasikan ditentukan sebagai bagian dari instruksi, yang secara langsung mengikuti opcode. Ini merupakan metode yang hampir (*straight forward*), dengan pemasukan informasi ke dalam memori yang sangat cepat. Namun demikian jika semua lokasi memori dialamatkan secara langsung, maka instruksi akan terdiri atas beberapa kata.
2. *Register addressing.* Dalam variant pengalamatan langsung ini, operand dilokasikan dalam register MPU dan alamat dari register merupakan bagian dari instruksi.

3. *Indirect addressing.* Alamat yang ditentukan dalam ruangan alamat dari instruksi akan mengambil lokasi dimana alamat yang efektif ditempatkan. Jika suatu lokasi adalah register MPU, maka mode pengalamatan ini disebut dengan *register indirect addressing*. Pengalamatan yang tidak langsung dapat dikembangkan pada beberapa level, yang disebut dengan *level of deferral*, dimana satu lokasi mengacu ke lokasi yang lain, yang sebaliknya akan mengacu ke lokasi yang lain, dan seterusnya.
4. *Base addressing.* Pendekatan ini digunakan pada pokoknya untuk referensi array atau untuk memberi lokasi kembali suatu program dalam memori. Alamat ini dibentuk dengan penambahan isi lokasi memori atau register (baik yang dapat dialamatkan tidak langsung) untuk menentukan angka yang disebut dengan **displacement**.
5. *Indexed addressing.* Alamat yang ditentukan dalam instruksi ditambahkan pada isi dari *index register* untuk membentuk alamat yang efektif. Dengan menggunakan instruksi khusus, register indeks dapat menambah atau mengurangi untuk memberikan fasilitas serangkaian melalui set konsekutif atau alamat yang diberi ruang. Jika register indeks secara otomatis meningkatkan atau mengurangi setiap waktu yang digunakan, maka proses ini disebut dengan **autoindexing**.
6. *Relative addressing.* Sangat berhubungan dengan indexed addressing, alamat yang ditentukan dalam instruksi ditambahkan pada isi program counter untuk membentuk alamat yang efektif.
7. *Immediate addressing.* Operand untuk suatu instruksi merupakan bagian dari instruksi, dan oleh karena itu, tidak ada pengalamatan yang diperlukan untuk mendapatkan informasi. Secara formal, immediate addressing bukanlah suatu mode pengalamatan.
8. *Page addressing.* Jika suatu angka dari memori utama adalah lebih besar dibandingkan dengan angka lokasi yang dapat di alamatkan *secara langsung* dengan suatu instruksi, maka memori utama akan dibagi ke dalam pages. Ukuran tiap page adalah sama dengan angka maksimum dari lokasi yang dapat dialamati secara langsung. Page dimasukkan dengan menggunakan salah satu mode pangalamatan yang diberikan daftarnya di sini, dan alamat yang efektif dari lokasi memori yang direferensikan merupakan alamat yang ditentukan untuk memulai page dimana muncul suatu instruksi.

Beberapa mode pengalamatan dapat diimplementasikan dengan menggunakan kombinasi dari daftar tersebut. Misalnya, suatu mode *indirect relaitve addressing*, alamat yang ditentukan dari suatu instruksi akan berada pada lokasi isi dimana akan ditambahkan pada isi program counter untuk membentuk alamat yang efektif dimana operandnya ditempatkan.

11.7. PEMBUATAN PENGATURAN WAKTU

Untuk memahami bagaimana mikroprosesor beroperasi, ini perlu mengenal dengan adanya pembuatan system **timing**. Pada pokoknya, MPU merupakan **synchronous sequential circuit** (lihat bagian 3.5), dimana teks berikutnya ditentukan dengan instruksi, dengan adanya state control logic unit (CLU), dan kadang-kadang dengan kejadian eksternal seperti interup. Seperti yang kita ketahui, bahwa sirkuit sekuensial dibandingkan dengan bagian memori dan bagian kombinasi (lihat Gambar 3-5). Bagian memori dalam MPU terdiri dari berbagai variasi register dan flag. Perubahan dalam isi register tersebut dan flag tersebut ditentukan dengan sirkuit kombinasi dalam MPU dan di-sinkron-kan dengan akurasi hitungan jam. Bagian kombinasi menunjukkan berbagai variasi fungsi yang berhubungan dengan, atau misalnya, manipulasi data, transfer data antarregister, pemeliharaan status, dan cariasi proses pembuatan keputusan, dan berlaku juga pada signal eksternal yang diterima dari memori dan bagian I/O.

Bagian kombinasi merupakan bagian yang berisi beberapa *gate level*, yang memperkenalkan jumlah significant dari waktu delay. Oleh karena itu, rata rata jam ditentukan sehingga ada waktu yang sesuai antara hitungan jam dengan akomodasi delay tersebut, termasuk juga waktu yang diperlukan untuk bagian memori untuk mengubah state.

Waktu yang diperlukan untuk pengambilan dan pembuatan istruksi disebut dengan **instruction cycle**. Ini merupakan waktu yang digunakan oleh MPU untuk mengambil opcode dari instruksi mikro (berikutnya) dari memori dan men-decodenya, untuk mengambil sisa instruksi (pengambilan operand) jika diperlukan, dan untuk membuat suatu instruksi. Lingkaran instruksi lebih lanjut dibagi ke dalam **machine cycles**, M_1, M_2, ... Suatu lingkaran mesin diperlukan setiap saat MPU memasukkan bagian eksternal, seperti memori atau bagian I/O. Dengan demikian untuk pembuatan suatu instruksi, MPU memberikan initial serangkaian lingkaran mesin. Tiap lingkaran mesin ini lebih lanjut dibagi ke dalam **clock state**, t_1, t_2, dan seterusnya ..... Meskipun angka jam ditentukan per variasi lingkaran mesin, setiap

jam mempunyai panjang yang sama, dan durasinya sama dengan periode jam. Oleh karena itu state jam menunjukkan unit terkecil dari aktifitas pemrosesan.

Untuk memasukkan bagian eksternal, maka MPU harus mengikuti kejadian seperti berikut ini:

1. Mengirimkan alamat suatu bagian pada alamat bus.
2. Mempersiapkan data kata dengan membuat signal kontrol *read* (*write*) yang menyebabkan suatu bagian untuk ditempatkan data katanya pada data bus. Jika suatu instruksi diambil, maka MPU akan mengambil kata pertamanya dan meningkatkan program counter (PC).
3. Membaca data atau instruksi kata dari data bus dan menempatkannya dalam register internal. Jika MPU merupakan mode write, maka bagian eksternal akan membaca kata dari data bus.

Dengan demikian minimal tiga state clock yang diperlukan untuk setiap lingkaran mesin. State tambahan yang diperlukan, misalnya, jika bagian eksternal tidak siap, untuk suatu decode instruksi, atau dalam lingkaran mesin yang menangani interupsi.

11.8. SINGLE-SLICED MIKROPROSESOR

Mikroprosesor single-chip adalah terbatas dan fleksibel untuk beberapa aplikasi. Misalnya, panjang kata dan rangkaian instruksi dari mikroprosesor telah ditentukan, dan kecepatan efektifnya juga terbatas. Beberapa aplikasi memerlukan kecepatan yang lebih tinggi atau instruksi khusus. Misalnya, sistem penerimaan data realtime, memerlukan konstruksi yang menunjukkan suatu perkalian, pembagian, dan aritmatika floating-point. Untuk menggunakan mikroprosesor sebagai pengontrol disk, kita memerlukan suatu instruksi seperti deteksi karakter dan penyisipan dan disk read/write. Kita dapat memprogram mikroprosesor single-chip untuk membuat instruksi khusus ini, tetapi akan menghasilkan dengan kecepatan yang rendah.

Untuk membuat mikroprosesor sangat cepat, kita memerlukan elemen logika yang mempunyai karakteristik switching sangat cepat. Integrated circuit (IC) keluarga logika dibuat dengan teknologi yang *populer* yang mampu dengan kecepatan efektif yang lebih besar, dengan minimal suatu faktor 4, dari pada yang disediakan oleh keluarga logika yang berdasarkan pada teknologi MOS. Namun

demikian bagian populer memerlukan tenaga relatif besar yang boros dan tidak dapat digabungkan ke dalam rancangan mikroprosesor single-chip.

Problem ini dapat diatasi dengan menggunakan teknik **bit-slice** untuk membuat mikroprosesor. Mikroprosesor bit-slice disusun dengan tingkat penampilan yang tinggi yang mampu menyediakan mikroprosesor single-chip. Mikroprosesor single-chip berbeda dengan mikroprosesor MOS konvensional dalam penanganan bagan data yang terpisah dari fungsi kontrol. MPU merupakan bagian dalam serangkaian chip LSI dan ditempatkan pada single chip.

Bagian kontrol dari bit-slice MPU adalah dapat diprogram dan berisi pengontrol program mikro dan memori. Bagian penanganan data berisi angka slice processor yang sama yang dihubungkan secara paralel. Setiap slice berisi angka kecil dari bit (biasanya dua atau empat) dari setiap register dan hubungan ALU bersama dengan sirkuit logika yang diperlukan untuk menunjukkan suatu operasi aritmatika dan logika dalam bit tersebeut. Dengan demikian, dengan menggunakan pendekatan bit-slice, para perancang dapat mengimplementasikan arsitektur mikroprosesor dengan berbagai variasi panjang kata dan serangkaian instruksi.

Untuk menunjukkan suatu operasi yang berguna yang sama dengan instruksi bahasa mesin tunggal, MPU harus berjalan melalui serangkaian instruksi mikro. Dalam mikroprosesor single-chip ini, dikerjakan dengan kontrol logik unit (CLU), dimana program mikronya biasanya tidak dapat dimasukkan ke pemakai. Sebaliknya, mikroprosesor bit-slice adalah: **user micro-programmable**. Pemakai harus membuat program mikro, menentukan berbagai variasi mikro routine, dan menyimpannya ke dalam ROM.

Pembuatan block khusus dari mikroprosesor bit-slice berisi, misalnya, 4-bit slice yang terdiri atas 4-bit ALU, sebuah shifter (yang mungkin merupakan bagian dari ALU), multiplexer, buses, register dan flag. Kita kemudian dapat menghubungkan block tersebut secara paralel untuk menangai panjang kata yang lebih besar. Misalnya, Gambar 11-4 menunjukkan hubungan 4-bit slice untuk membentuk 16-bit MPU. Ini penting memperhatikan dua benda dalam relasi organisasi ini:

1. Data dan alamat bus dibuat hingga 16-bit dari 4 bit setiap slice.
2. Kata instruksi mikro yang diambil dari memori program mikro disobek. Satu kelompok instruksi mikro bit menyediakan opcode instruksi mikro pada seluruh slice-nya. Kelompok lain dari bit akan berjalan ke pengontrol program mikro untuk menentukan, misalnya, alamat instruksi mikro selanjutnya.

Ini jelas bahwa penambahan komponen diperlukan untuk mengimplementasikan suatu mikroprosesor yang lengkap dari slice. Komponen ini termasuk suatu counter program, register instruksi dan decoder, register untuk menangani I/O dan interfacing memori utama, dan mungkin stack. Berbagai variasi yang mendukung chip ada untuk menangani pembuatan suatu instruksi, tetapi sirkuit logika tambahan hampir selalu diperlukan untuk membuat mikroprosesor yang lengkap dari slice. Kekurangan lainnya dari mikroprosesor bit-slice adalah terbatas kemampuannya dari dukungan perangkat lunak. Namun demikian, kekurangan ini harus dikontraskan dengan beberapa keuntungan lain yang disediakan dengan pendekatan arsitektural ini. Mikroprosesor bit-slice menawarkan suatu rancangan yang lebih fleksibel dibandingkan dengan mikroprosesor konvensional dan dapat menyesuaikan dengan aplikasi yang baru. Slice dapat dengan mudah dihubungkan untuk mencapai suatu tingkat tertentu dari data dengan kecepatan tinggi, dan perancang dapat menentukan panjang kata dan serangkaian instruksi.

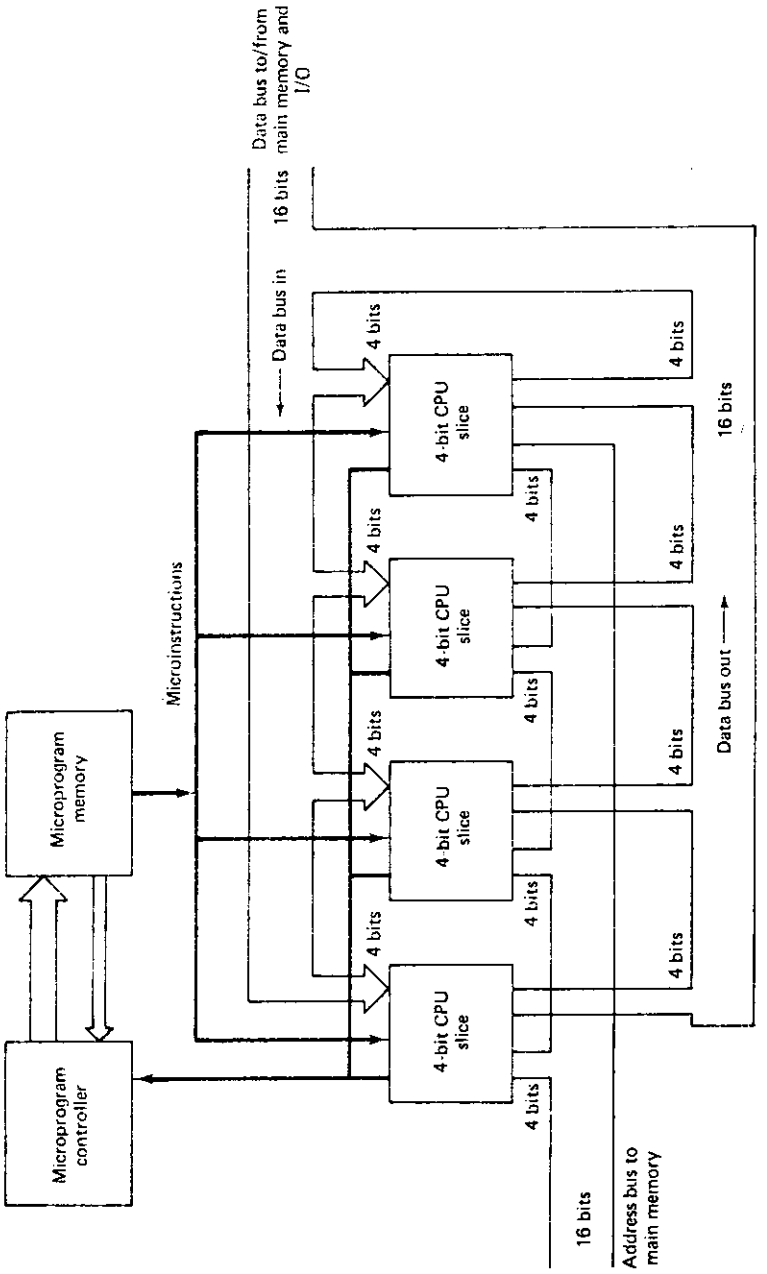
Organisasi Bit-slice

Mikroprosesor bit-slice yang dapat diprogram berisi dua modul pokok: yaitu modul kontrol dan modul pemrosesan data. Dua elemen pokok dari **control module** adalah **microprogram controller** (atau **sequencer**) dan **microprogram memory**. Organisasi CPU semacam ini ditunjukkan secara skematis dalam Gambar 11-5, yang mengilustrasikan hubungan dari elemen-elemen tersebut.

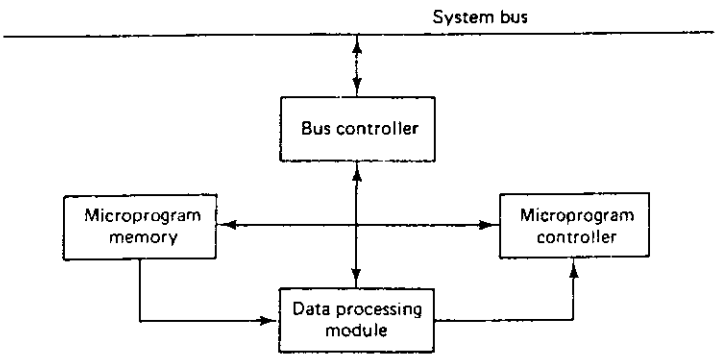
Data processing module, mencakup ALU, register dengan tujuan umum, program counter, status register, register instruksi, register data, register alamat memori, dan sirkuit logika yang diperlukan untuk menunjukkan operasi aritmatika pokok dan logika. Modul pemrosesan data ini dikontrol oleh mikroroutine, jumlah totalnya disebut dengan *microprogram*, yang disimpan dalam memori program mikro. Untuk mencapai kecepatan tinggi, memori ini biasanya dibuat dari bipolar ROM atau PROM.

Kontroler mikroprogram merangkai suatu pesan dimana instrusksi mikro dibuat. Setiap instruksi mikro memberi awalan aksi dalam elemen pemrosesan data dan atau **bus controller**. Instruksi mikro terdiri atas angka field yang mengontrol operasi elemen pemrosesan data, menyediakan untuk operasi seperti kontrol I/O, dan menentukan instruksi mikro selanjutnya untuk dibuat dalam basis informasi yang diperoleh dari opcode dan kontrol flag dari instruksi makro.

Kata instruksi mikro yang diambil dari memori program mikro dibagi ke dalam kelompok yang berbeda. Kelompok yang pertama diumpanbalikkan pada



Gambar 11-4 MPU 16-bit dibentuk dari empat 4-bit slice.



Gambar 11-5 Diagram skema CPU bit-slice.

kontroler program mikro. Kelompok yang kedua akan mengontrol modul pemrosesan data. Kelompok yang ketiga akan mengatur aliran informasi antara kontroler bus dan CPU. Prosedur ini disebut dengan **horizontal microprogramming** dan memungkinkan beberapa fungsi untuk ditunjukkan secara simultan dalam lingkaran instruksi mikro tunggal. Namun demikian, jika tidak semua signal kontrol diperlukan dalam lingkaran instruksi mikro yang sama, maka pendekatan ini sangat sia-sia karena setiap bit kontrol ditunjukkan pada lingkaran tunggal dan dibuat setiap lingkaran instruksi mikro baik diperlukan atau tidak. Pada khususnya, pendekatan ini menuntun kata instruksi mikro yang lebih panjang.

Suatu alternatif pendekatan pada horisontal programming adalah **timeshare** signal kontrol yang secara mutual lain dengan lingkaran instruksi mikro yang sama dan mengurangi angka bit kontrol. Prosedur ini disebut dengan **vertical programming**. Untuk mengilustrasikan perbedaan anatara dua pendekatan ini, perhatikan situasi dimana tujuh signal kontrol dibuat dalam horisontal programming tetapi hanya satu sebenarnya yang diperlukan dalam lingkaran instruksi mikro yang sama. Jika suatu signal secara mutual adalah eksklusif, kita dapat menggunakan vertikal programming untuk membuat salah satu dari tujuh signal kontrol dengan hanya tiga kontrol bit dengan menggunakan decoder 3 X 8 secara sederhana.

SOAL

1. Tunjukkan secara sistimatika bagaimana MPU mengguńakan interup vector dalam data bus untuk cabang bagian I/O interup service routine dalam memori utama.
2. Perhatikan 8-bit MPU yang mempunyai 16-bit alamat bus. Gabungan dengan MPU adallah 256 bagian I/O.
   (a). Gunakan sebanyak 4 x 16 decoder chip — masing-masing dengan input chip enable (E) — yang diperlukan untuk membuat suatu sirkuit, dalam bentuk diagram block, yang akan membuat 256 bagian I/O memilih signal. Sirkuit harus didapatkan dengan baris kontrol I/O (IOC) dari MPU.
   (b). Berapa banyak alamat bit yang diperlukan?
   (c). Porsi mana dari 16-bit alamat bus yang membawa bagian alamat?
3. Berapa angka minimum dari alamat bit yang diperlukan untuk alamat unit memori yang disusun atas dua 256 x 8-bit memori chip?. Tunjukkan diagram blok yang mengimplementasikan unit memori ini.
4. Berapa angka minimum dari alamat bit yang diperlukan untuk alamat unit memori yang disusun atas empat 256 x 8-bbitmemori chip?
   (a). Berapa bit yang ditempatkan untuk fungsi pemilihan chip?
   (b). Berapa bit yang ditempatkan untuk fungsi alamat kata memori?
   (c). Tunjukkan diagram block yang mengimplementasikan unit memori ini.
5. Berapa bagian yang dapat dipilih dengan 8-bit alamat dengan menggunakan decoder 8 x 256? Berikan code heksadesimal dari alamat 8-bit :
   (a). Kapan pemilihan bagian angka 1.
   (b). Kapan pemilihan bagian angka 10.
   (c). Kapan pemilihan bagian angka 65.
6. Berapa signal kontrol yang dapat dibuat dari 9 instruksimikro bit dengan menggunakan tiga decoder 3 x 3? Tunjukkan implementasi diagaram blocknya.
7. Perhatikan 16-bit MPU dimana instruksinya disusun atas 8-bit field. Byte pertama menunjukkan opcode, dan bit kedua menunjukkan alamat field atau operand.
   (a). Berapa maksimum spasi memori yang dapat dialamatkan?
   (b). Berapa banyak bit yang diperlukan untuk progrma counter (PC) dan untuk register instruksi (IR).

(c). Apa (impact) pada kecepatan MPU jika mikroprosesor mempunyai (i) 8-bit alamat bus dan 8-bit data bus, atau (ii) 8-bit alamat bus dan 4-bit data bus?.

8. Suatu mikroprosesor mempunyai 332 instruksi dan 16 K kata dari penyimpanan memori utama. Berapa banyak bit yang diperlukan untuk kata instruksi dalam system single-address (lihat Bab 5).

9. Sediakan flowchart yang menunjukkan operasi yang dilakukan oleh MPU dalam pengambilan dan pembuatan instruksi program.

10. Anggap bahwa accumulator (ACC) berisi angka $(C6)_{16}$ register B berisi angka $(94)_{16}$, dan flag pembawa C berisi 1. Baik ACC dan B adalah 8-bit register. Apa isi dari ACC dan C setelah pembuatan setiap instruksi berikut ini?

(a) ADD B ACC ← (ACC) + (B)

(b) XOR B ACC ← (ACC) + (B)

(c) CMA ACC ← (ACC)

(d) RTL Rotate ACC dan C left

(e) SUB B ACC ← (ACC) - (B)

11. Perhatikan suatu instruksi yang disebut dengan ADT yang mengambil suatu operand dari memori, menambahkannya ke dalam akumulator, dan membaginya dengan hasil 2. Bagian fraksional hasilnya diabaikan.

(a). Berapa state jam yang diperlukan untuk membuat instruksi tersebut?

(b). Gunakan register bahasa transfer (RTL; lihat Bab 4) untuk menunjukkan serangkaian operasi mikro yang diperlukan untuk membuat instruksi ini selama setiap state jam.

12. Perhatikan dua MPU, A dan B. MPU A memerlukan 4 jam state untuk memasukkan memori dan 2 jam state untuk menunjukkan suatu penambahan. MPU B memerlukan 5 jam state untuk memasukkan memori dan 2 jam state untuk menunjukkan suatu penambahan.

(a). Jika MPU A ditangani oleh 4-MHz jam dan MPU B ditangani oleh 6MHz jam, berapa total waktu yang diperlukan setiap mikroprosesor untuk mengambil tiga angka dari memori, jumlahkanlah, dan simpan hasilnya kembali dalam memori.

(b). Jika MPU A menunjukkan operasi dalam bagian (a) dalam waktu yang sama yang digunakan oleh MPU B, menjadi bagaimana frekuensi jam?.

13. Tiga status bit s_0, s_1, dan s_2 digunakan oleh MPU untuk menunjukkan transaksi bus yang diperlukan seperti ditunjukkan dalam tabel :

s_2	s_1	s_0	Bus Cycle
0	0	0	Interrupt acknowledge (INRA)
0	0	1	I/O read (IOR)
0	1	0	I/O write (IOW)
0	1	1	Halt
1	0	0	Fetch instruction (FI)
1	0	1	Memory read (MR)
1	1	0	Memory write (MW)
1	1	1	Inactive

(a). Rancangan sirkuit logika untuk decode status informasi untuk pengontrol bus.

(b). Berikan contoh transaksi yang diperlukan MPU untuk menjadikan bus tidak aktif.

14. Gambar 11-2 (b) menunjukkan signal waktu dari jam dua-phase. Rancangan sirkuit yang menghasilkan signal ketika ditangani oleh sumber hitungan untuk cycle duty berikut ini :

(a). 25 %.

(b). 50 %.

(c). 75 %.

Plot diagram waktu yang menunjukkan hubungan antara phase 1, phase 2, dan sumber hitungan unstuk setiap kasus. (Perhatikan; Duty cycle berarti prosentasi waktu dimana signal dari sumber hitungan pada logika 1). (Perhatikan: Gunakan T flip-flop).

15. (a). Selalu diperlukan untuk poll bagian I/O untuk mengubah data dengan MPU ketika I/O terprogram digunakan?

(b). Jika dua bagian telah dilayani pada waktu yang sama, bagaimana MPU memilih yang harus dilayani pertama kali di bawah program I/O dalam interup?

16. Seperti diilustrasikan dalam Gambar 11-4, MPU slice dapat dikurangi untuk mencapai suatu tingkat yang sesuai untuk angka yang dioperasikannya. Namun demikian, pengurangan tidak merupakan satu satunya jalan dimana bit-slice MPU dapat dikombinasikan untuk menunjukkan pemrosesan

paralel. Anggap bahwa kita ingin memproses vector tiga demensi, setiap 8-bit panjangnya. Dengan menggunakan 4-bit slice seperti dalam Gambar 11-4, tunjukkan ubungan slice tersebut untuk membentuk unit hitungan vector tiga demensi.

17. Pada berbagai fungsi waktu, MPU kadang-kadang menggunakan timer interval. Termasuk down counter (lihat Bab 3) yang dipanggil dengan angka dari suatu bagian output, dan jika interval telah hilang (counter mencapai nol), maka flag interup akan diatur. Jika suatu interupsi telah diselesaikan, alamat permulaan dari service routine ditempatkan pada data bus. Anggap bahwa 8-bit bus, dan rancangan interval timer dalam bentuk diagram block.