Startup pada Sistem

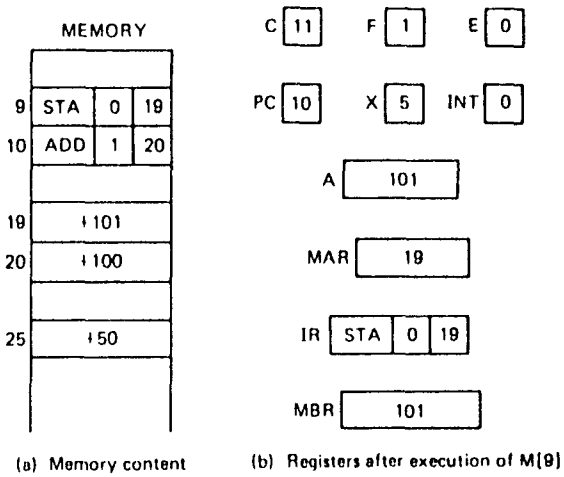
Setiap komputer pasti memiliki cara-cara tertentu untuk memulai segala sesuatunya sewaktu ia dinyalakan. Hal ini biasanya termasuk inisialisasi register-register tertentu dan kemudian menjalankan instruksi-instruksi tertentu yang menangani inisialisasi tersebut selebihnya. Instruksi-instruksi khusus ini dapat berupa *resident* (menduduki) pada bagian memori tertentu atau secara otomatis dibaca ke dalam memori dari piranti I/O khusus.

Pada mesin SIC, penyalaan sistem menyebabkan S dan F di-*set* bernilai 1; E dan C dibersihkan; SW[ICODE] di-*set* pada "startup"; dan PC di-*set* bernilai 1. Hal ini menyebabkan CPU langsung menjalankan routine *interrupt handler* yang resident, yang mulai mengintepretasikan interupsi startup. Tambahan bagi proses startup; sistem, diperlukan juga agar dapat me-*reset* sistem tanpa mematikannya dan kemudian menyalakannya kembali. Pada SIC hal ini dapat dilakukan dengan menggunakan **saklar start/stop** manual. Dengan mengubah flip-flop mekanis menjadi 1, akan menyebabkan sistem menjalani routine power-up. Memindahkan saklar ke 0 hanya membersihkan S, dengan demikian akan memberhentikan counter dan memberhentikan seluruh mesin.

4.4 PELACAKAN EKSEKUSI PADA SEBUAH INSTRUKSI SIC

Setelah menjabarkan siklus CPU, mekanisme pemilihan waktu dan pengendalian serta unit-unit fungsional pada SIC, sebenarnya kita telah mendefinisikan bagaimana register dan unit-unit pada komputer saling berhubungan dan bagaimana data ditranfer di antara mereka. Pada bagian ini, kita akan melacak suatu instruksi melalui keseluruhan siklus fecth dan eksekusinya sebagai sebuah contoh tentang cara kerja SIC.

Untuk memulainya, anggaplah bahwa isi unit memori adalah seperti yang ditunjukkan pada Gambar 4-6(a). Kita akan melacak melalui pelaksanaan instruksi yang sekarang tersimpan dalam M[10]. Kita anggap instruksi sebelumnya, M[9], telah dijalankan dan isi dari register CPU adalah seperti yang ditunjukkan pada Gambar 4-6(b). Perhatikan bahwa flag F dan E telah disusun untuk siklus fetch berikutnya, tetapi counter C belum kembali ke 00.

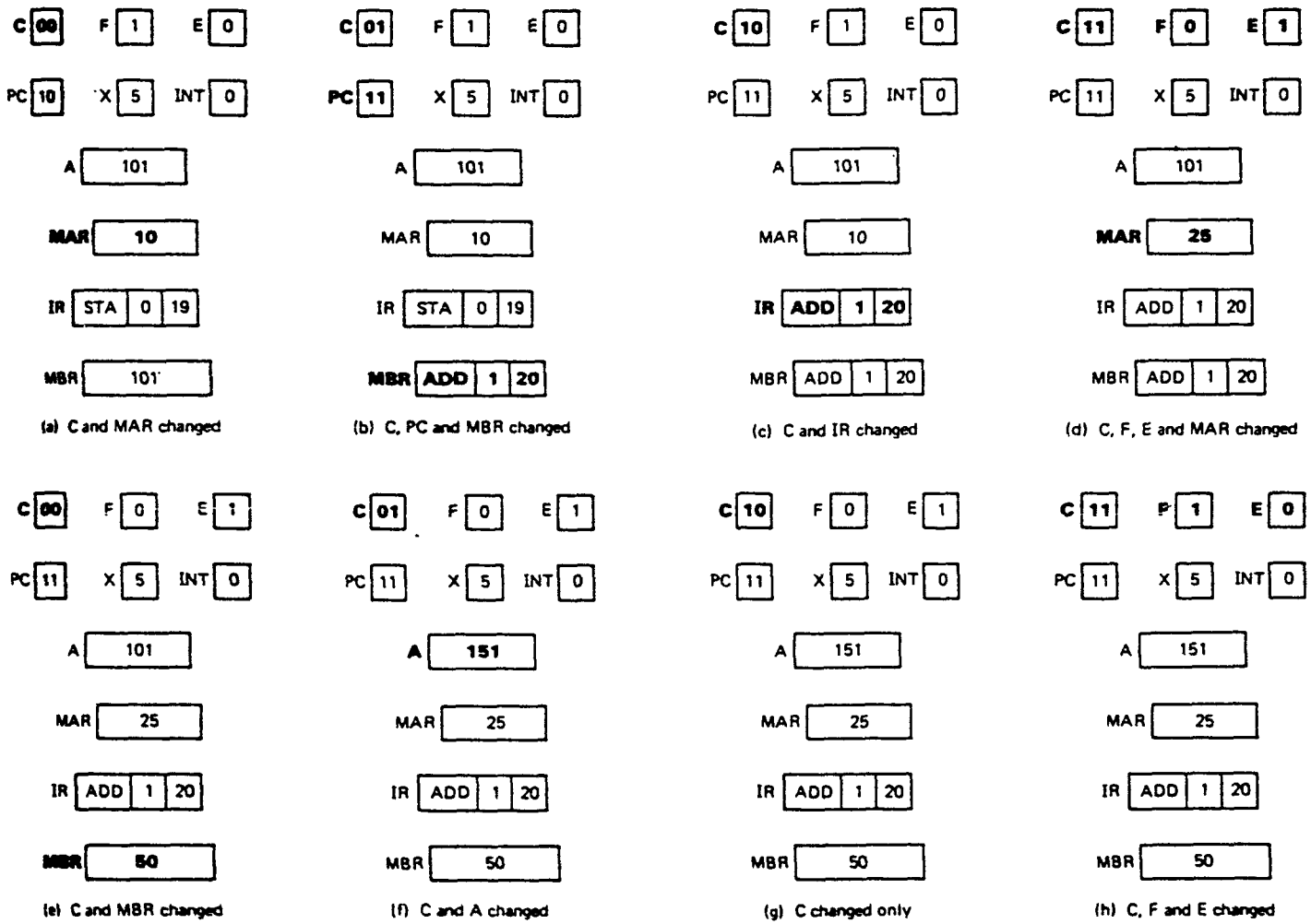


Gambar 4-6 Isi memori inisial dan register

Eksekusi tanpa Interupsi

Pelacakan instruksi pada M[10] dilukiskan dalam rangkaian "*snapshot*" pada Gambar 4-7, dimana setiap snapshot menunjukkan keadaan dari sistem tepat setelah C ditingkatkan. Register yang diubah sejak snapshot sebelumnya ditunjukkan dengan huruf tebal.

Ketika counter bergerak ke 00, akan dihasilkan sinyal waktu pertama, t_0, pada siklus fetch. Berdasarkan instruksi siklus fetch yang diberikan dalam Tabel 4.6, aksi-aksi berikut ini akan dilakukan. Pada saat t_0, isi PC akan ditransfer ke MAR [Gambar 4-7(a)]. Pada saat t_1, word memori yang dijabarkan oleh MAR dibaca ke dalam MBR dan PC meningkat [bagian (b)]. Pada sinyal waktu ketiga, t_2, isi MBR disalin ke IR [bagian (c)]. Transfer yang terjadi pada sinyal waktu terakhir siklus fetch, t_3, tergantung pada nilai field IX pada IR. Karena nilai tersebut adalah 1, maka nilai field AD pada IR ditambahkan ke nilai register X dan disimpan dalam MAR. Sebagai tambahan; flag F dibersihkan dan flag E di-*set*. Transfer ini ditunjukkan pada Gambar 4-7(d). Sewaktu counter C kembali ke 00, maka flag F dan E akan memaksa CPU untuk memasuki siklus eksekusi. Siklus eksekusi yang akan digunakan dikendalikan oleh opcode yang saat itu berada dalam field OP pada IR. Dalam hal ini, dipakai siklus eksekusi ADD yang



Gambar 4-7 Isi register selama eksekusi tanpa interupsi

ditunjukkan sebelumnya pada Tabel 4.8. Dengan demikian pada sinyal t_0, isi word memori yang dijabarkan oleh MAR, M[25], disalin ke dalam MBR [bagian (e)].

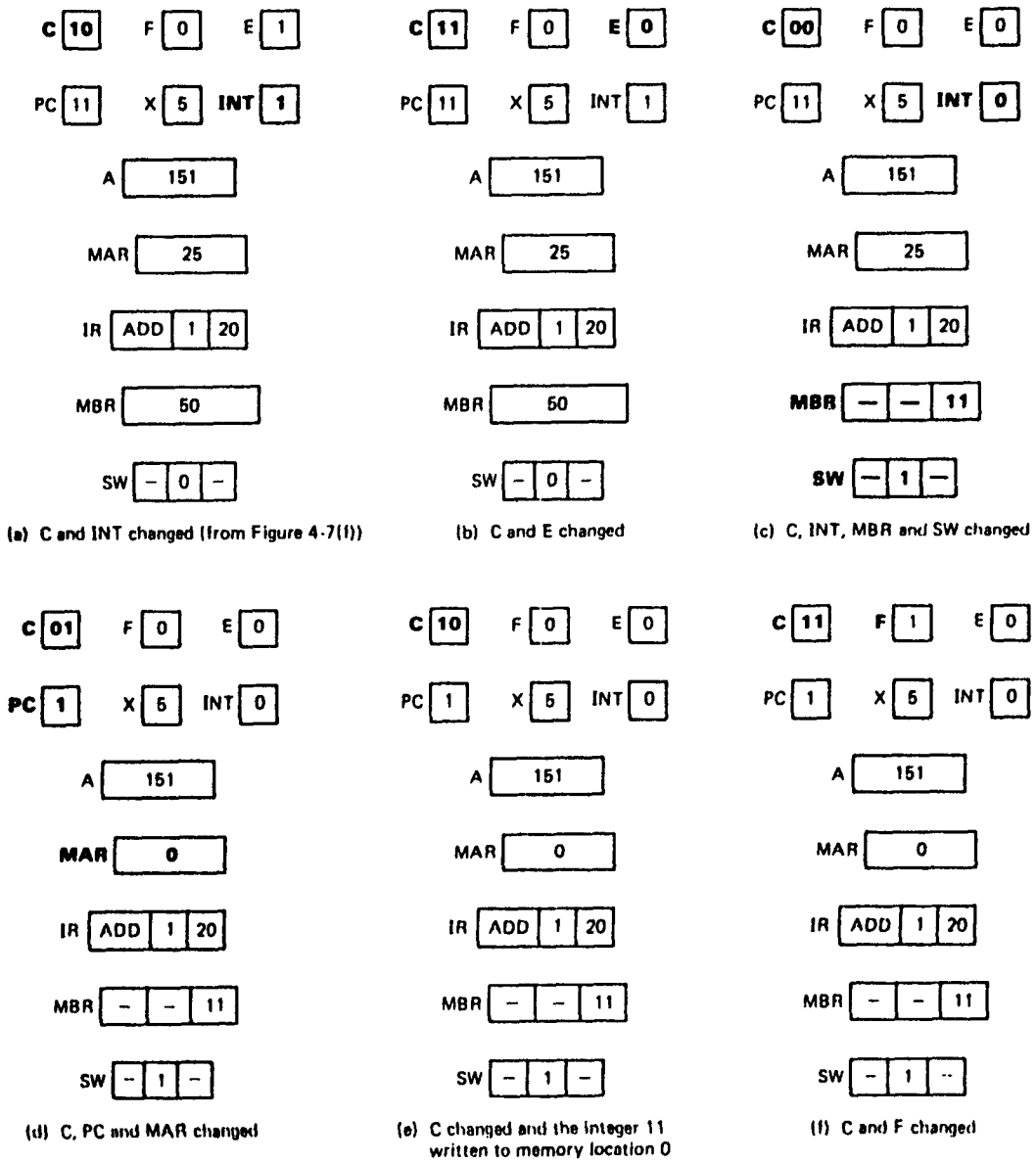
Pada saat t_1, nilai MBR ditambahkan ke A dan juga disimpan ke A. Karena M[25] berisi 50, nilai A menjadi 151 [bagian (f)]. Tidak ada tindakan yang terjadi pada sinyal waktu t_2, selain peningkatan C secara otomatis [bagian (g)]. Pada contoh pertama ini, tidak ada interupsi yang terjadi selama siklus fetch maupun siklus eksekusi. Dengan demikian pada Gambar 4-7(h), F dan E di-susun untuk siklus fecth berikutnya.

Eksekusi dengan Interupsi

Sekarang, mari kita anggap bahwa kadang-kadang selama siklus fecth maupun siklus eksekusi pada Gambar 4-7 terjadi interupsi. Kita ambil pelacakan instruksi dalam M[10] pada t_2, F = 0 dan E = 1 [Gambar 4-7(g) pada contoh terdahulu], tetapi dengan INT = 1 dan SW[MASK] = 0. Gambar 4-8(a) melukiskan keadaan baru ini. Pada saat ini, sewaktu sinyal waktu terakhir siklus eksekusi ADD, t_3, (NOT SW[MASK] AND INT) bernilai benar, sehingga kondisi pernyataan IF menjadi salah. Sebagai hasilnya, hanya E yang berubah [Gambar 4-8(b)].

Ketika C meningkat menjadi 00, flag E dan F akan mengarahkan CPU ke siklus interupsi (ditunjukkan dalam Tabel 4.7). Pada sinyal waktu pertama pada siklus ini, t_0, akan terjadi tiga transfer. Isi PC ditransfer ke field AD pada MBR, flag INT dibersihkan dan field MASK pada SW di-*set* [bagian (c)]. Pada saat t_1, PC di-*set* bernilai 1 dan MAR dibersihkan [bagian (d)]. Kemudian pada sinyal waktu ketiga, t_2, MBR ditulis ke memori pada address 0 [bagian (e)]. Tindakan terakhir pada siklus ini adalah menyusun F dan E untuk siklus fetch. Hal ini dikerjakan dengan men-*set* F ke 1 seperti ditunjukkan pada Gambar 4-8(f).

Seperti yang dapat dilihat pada kedua contoh tersebut, baik terjadi interupsi maupun tidak, pada akhir suatu siklus instruksi yang lengkap, CPU sudah siap untuk mulai bekerja pada instruksi berikutnya. Sebenarnya, instruksi mana yang berikutnya tergantung pada apa yang terjadi selama siklus eksekusi sebelumnya dan jika siklus interupsi telah dilaksanakan. Bagaimanapun juga hal ini tidak relevan untuk CPU yang hanya terus bekerja dan mengambil instruksi yang direferensikan oleh PC. Lihat kembali Gambar 4-7 dan 4-8 untuk lebih memahami cara kerja SIC.



Gambar 4-8 Isi register selama eksekusi dengan interupsi