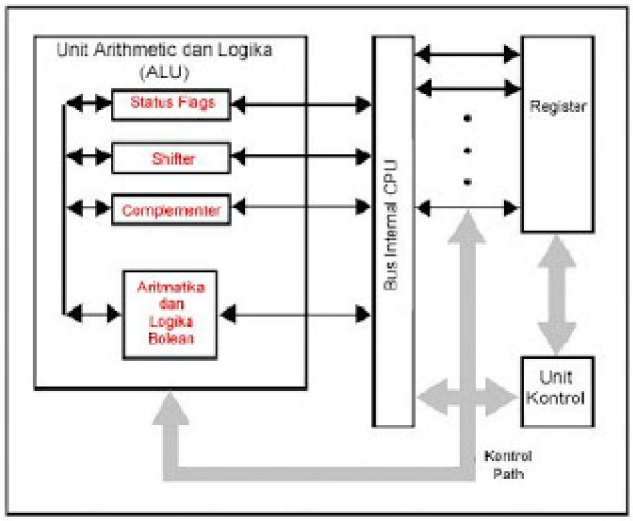
PERTEMUAN 13

STRUKTUR CPU

CPU merupakan komponen terpenting dari sistem komputer. CPU adalah komponen pengolah data berdasarkan instruksi – instruksi yang diberikan kepadanya. CPU terdiri dari dua bagian utama yaitu unit kendali (control unit) dan unit aritmatika dan logika (ALU). CPU atau Central Processing Unit dapat dikatakan juga otak dari komputer itu sendiri Sebuah komputer paling canggih sekalipun tidak akan berarti tanpa adanya CPU yang terpasang di dalamnya. Dalam kesehariannya CPU memiliki tugas utama untuk mengolah data berdasarkan instruksi yang ia peroleh.



CPU sendiri sebenarnya masih terbagi atas beberapa komponen yang saling bekerja sama untuk membentuk suatu unit pengolahan Disamping itu, CPU mempunyai beberapa alat penyimpan yang berukuran kecil yang disebut *register*.Terdapat empat komponen utama penyusun CPU, yaitu

1. Arithmetic and Logic Unit (ALU)
2. Control Unit
3. Registers
4. CPU Interconnections

- Arithmetic and Logic Unit (ALU)

Arithmetic and Logic Unit atau sering disingkat ALU saja dalam bahasa Indonesia kira-kira berarti Unit Logika dan Aritmatika. Bagian ini mempunyai tugas utama untuk membentuk berbagai fungsi pengolahan data komputer. Sering juga disebut sebagai bahasa mesin, karena terdiri dari berbagai instruksi yang menggunakan bahasa mesin. ALU sendiri juga masih terbagi menjadi dua komponen utama, yaitu

1. arithmetic unit (unit aritmatika), bertugas untuk menangani pengolahan data yang berhubungan dengan perhitungan, dan
2. boolean logic unit (unit logika boolean), bertugas menangani berbagai operasi logika.

- Control Unit

Control Unit atau Unit Kendali, mempunyai tugas utama untuk mengendalikan operasi dalam CPU dan juga mengontrol komputer secara keseluruhan untuk menciptakan sebuah

sinkronisasi kerja antar komponen dalam melakukan fungsinya masing-masing. Di samping itu, control unit juga bertugas untuk mengambil instruksi-instruksi dari memori utama dan menentukan jenis instruksi tersebut.

- CPU Interconections

CPU Interconnections merupakan sistem koneksi dan bus yang menghubungkan komponen internal CPU dengan bus-bus eksternal CPU.

- Sedangkan komponen eksternal CPU diantaranya

1. sistem memori utama,
2. sistem masukan/keluaran (input/output),
3. dan sistem-sistem lainnya.

1. FUNGSI CPU

CPU berfungsi seperti kalkulator , hanya saja CPU jauh lebih kuat daya pemrosesannya. Fungsi utama dari CPU adalah melakukan operasi aritmatika dan logika terhadap data yang diambil dari memori atau dari informasi yang dimasukkan melalui beberapa perangkat keras , seperti papan ketik ,pemindai , tuas kontrol , maupun tetikus . CPU dikontrol menggunakan sekumpulan instruksi perangkat lunak komputer . Perangkat lunak tersebut dapat dijalankan oleh CPU dengan membacanya dari media penyimpan, seperti cakram keras , disket , cakram padat , maupun pita perekam. Instruksi-instruksi tersebut kemudian disimpan terlebih dahulu pada memori fisik (RAM), yang mana setiap instruksi akan diberi alamat unik yang disebut alamat memori. Selanjutnya, CPU dapat mengakses data-data pada RAM dengan menentukan alamat data yang dikehendaki.

Saat sebuah program dieksekusi, data mengalir dari RAM ke sebuah unit yang disebut dengan bus , yang menghubungkan antara CPU dengan RAM. Data kemudian didekode dengan menggunakan unit proses yang disebut sebagai pendekoder instruksi yang sanggup menerjemahkan instruksi. Data kemudian berjalan ke unit aritmatika dan logika (ALU) yang melakukan kalkulasi dan perbandingan. Data bisa jadi disimpan sementara oleh ALU dalam sebuah lokasi memori yang disebut dengan register supaya dapat diambil kembali dengan cepat untuk diolah.

ALU dapat melakukan operasi-operasi tertentu, meliputi penjumlahan, perkalian, pengurangan, pengujian kondisi terhadap data dalam register, hingga mengirimkan hasil pemrosesannya kembali ke memori fisik , media penyimpan, atau register apabila akan mengolah hasil pemrosesan lagi. Selama proses ini terjadi, sebuah unit dalam CPU yang disebut dengan penghitung program akan memantau instruksi yang sukses dijalankan supaya instruksi tersebut dapat dieksekusi dengan urutan yang benar dan sesuai. Selain itu, Fungsi CPU juga untuk penjalankan program – program yang disimpan dalam memori utama dengan cara mengambil instruksi – instruksi, menguji instruksi tersebut dan mengeksekusinya satu persatu sesuai alur perintah. Untuk memahami fungsi CPU dan caranya berinteraksi dengan komponen lain, perlu kita tinjau lebih jauh proses eksekusi program. Pandangan paling sederhana proses eksekusi program adalah dengan mengambil pengolahan instruksi yang terdiri dari dua langkah, yaitu : operasi pembacaan instruksi (*fetch*) dan operasi pelaksanaan instruksi (*execute*).

A. Aksi CPU

- CPU – Memori, perpindahan data dari CPU ke memori dan sebaliknya.

- CPU – I/O, perpindahan data dari CPU ke modul I/O dan sebaliknya.
- Pengolahan Data, CPU membentuk sejumlah operasi aritmatika dan logika terhadap data.
- Kontrol, merupakan instruksi untuk pengontrolan fungsi atau kerja. Misalnya instruksi pengubahan urusan eksekusi.

Siklus fetch-eksekusi bisa dijelaskan sebagai berikut :

1. Di awal setiap siklus, CPU akan membaca dari memori utama,
2. Sebuah register, yang disebut Program Counter (PC), akan mengawasi dan menghitung instruksi selanjutnya,
3. Ketika CPU membaca sebuah instruksi, Program Counter akan menambah satu hitungannya,
4. Alu instruksi-instruksi yang dibaca tersebut akan dimuat dalam suatu register yang disebut register instruksi (IR), dan akhirnya
5. CPU akan melakukan interpretasi terhadap instruksi yang disimpan dalam bentuk kode binari, dan melakukan aksi yang sesuai dengan instruksi tersebut..

B. Siklus Eksekusi

Siklus eksekusi untuk suatu instruksi dapat melibatkan lebih dari sebuah referensi ke memori. Disamping itu juga, suatu instruksi dapat menentukan suatu operasi I/O. Perhatikan pada Gambar Diagram siklus intruksi.

C. Siklus instruksi

- *Instruction Addess Calculation (IAC)*, yaitu mengkalkulasi atau menentukan alamat instruksi berikutnya yang akan dieksekusi. Biasanya melibatkan penambahan bilangan tetap ke alamat instruksi sebelumnya. Misalnya, bila panjang setiap instruksi 16 bit padahal memori memiliki panjang 8 bit, maka tambahkan 2 ke alamat sebelumnya.
- *Instruction Fetch (IF),* yaitu membaca atau pengambil instruksi dari lokasi memorinya ke CPU.
- *Instruction Operation Decoding (IOD),* yaitu menganalisa instruksi untuk menentukan jenis operasi yang akan dibentuk dan operand yang akan digunakan.
- *Operand Address Calculation (OAC),* yaitu menentukan alamat operand, hal ini dilakukan apabila melibatkan referensi operand pada memori.
- *Operand Fetch (OF),* adalah mengambil operand dari memori atau dari modul I/O.
- *Data Operation (DO),* yaitu membentuk operasi yang diperintahkan dalam instruksi.
- *Operand store (OS),* yaitu menyimpan hasil eksekusi ke dalam memori.

2. FUNGSI INTERRUPT

Fungsi interupsi adalah mekanisme penghentian atau pengalihan pengolahan instruksi dalam CPU kepada routine interupsi. Hampir semua modul (memori dan I/O) memiliki mekanisme yang dapat menginterupsi kerja CPU. Tujuan interupsi secara umum untuk menejemen pengeksekusian routine instruksi agar efektif dan efisien antar CPU dan modul – modul I/O maupun memori. Setiap komponen – komputer dapat menjalankan tugasnya secara bersamaan, tetapi kendali terletak pada CPU disamping itu kecepatan eksekusi masing – masing modul berbeda sehingga dengan adanya fungsi interupsi ini dapat sebagai sinkronisasi kerja antar modul. Macam – macam kelas sinyal interupsi :

- *Program*, yaitu interupsi yang dibangkitkan dengan beberapa kondisi yang terjadi pada hasil eksekusi program. Contohnya: arimatika overflow, pembagian nol, oparasi ilegal.
- *Timer*, adalah interupsi yang dibangkitkan pewaktuan dalam procesor. Sinyal ini memungkinkan sistem operasi menjalankan fungsi tertentu secara reguler.
- *I/O*, sinyal interupsi yang dibangkitkan oleh modul I/O sehubungan pemberitahuan kondisi error dan penyelesaian suatu operasi.
- *Hardware failure,* adalah interupsi yang dibangkitkan oleh kegagalan daya atau kesalahan paritas memori.

3. TUJUAN INTERUPSI

- Secara umum untuk manajemen pengeksekusian routine instruksi agar efektif dan efisien antar CPU dan modul-modul I/O maupun memori.
- Setiap komponen computer dapat menjalankan tugasnya secara bersamaan, tetapi kendali terletak pada CPU disamping itu kecepatan eksekusi masing-masing modul berbeda.
- Dapat sebagai sinkronisasi kerja antar modul

4. KELAS SINYAL INTERUPSI

- Program, yaitu interupsi yang dibangkitkan dengan beberapa kondisi yang terjadi pada hasil eksekusi program. Contohnya : aritmatika overflow, pembagian nol, operasi ilegal.
- Timer, adalah interupsi yang dibangkitkan perwaktuan dalam processor. Sinyal ini memungkinkan sistem operasi menjalankan fungsi tertentu secara reguler.
- I/O, sinyal interupsi yang dibangkitkan oleh modul I/O sehubungan pemberitahuan kondisi error dan penyelesaian suatu operasi.
- Hardware failure, adalah interupsi yang dibangkitkan oleh kegagalan daya atau kesalahan paritas memori.

5. PROSES INTERUPSI

- Dengan adanya mekanisme interupsi, procesor dapat digunakan untuk mengeksekusi instruksi-instruksi lain.
- Saat suatu modul telah selesai menjalankan tugasnya dan siap menerima tugas berikutnya, maka modul ini akan mengirimkan permintaan interupsi ke procesor.
- Kemudian procesor akan menghentikan eksekusi yang dijalankannya untuk menghandle routine interupsi.
- Setelah program interupsi selesai, maka procesor akan melanjutkan eksekusi programnya.
- Saat sinyal interupsi diterima procesor ada dua kemungkinan tindakan, yaitu interupsi diterima/ditolak dan interupsi ditolak.

Register procesor

Register procesor, dalam arsitektur komputer , adalah sejumlah kecil memori komputer yang bekerja dengan kecepatan sangat tinggi yang digunakan untuk melakukan eksekusi terhadap program-program komputer dengan menyediakan akses yang cepat terhadap nilai-nilai yang umum digunakan. Umumnya nilai-nilai yang umum digunakan adalah nilai yang sedang dieksekusi dalam waktu tertentu.

Register procesor berdiri pada tingkat tertinggi dalam hierarki memori : ini berarti bahwa kecepatannya adalah yang paling cepat; kapasitasnya adalah paling kecil; dan harga tiap bitnya adalah paling tinggi. Register juga digunakan sebagai cara yang paling cepat dalam sistem komputer untuk melakukan manipulasi data . Register umumnya diukur dengan satuan bit yang dapat ditampung olehnya, seperti "register 8-bit", "register 16-bit", "register 32-bit", atau "register 64-bit" dan lain-lain. Istilah register saat ini dapat merujuk kepada kumpulan register yang dapat diindeks secara langsung untuk melakukan input/output terhadap sebuah instruksi yang didefinisikan oleh set instruksi . untuk istilah ini, digunakanlah kata "Register Arsitektur". Sebagai contoh set instruksi Intel x86 mendefinisikan sekumpulan delapan buah register dengan ukuran 32-bit, tapi CPU yang mengimplementasikan set instruksi x86 dapat mengandung lebih dari delapan register 32-bit.

1. JENIS REGISTER

Register terbagi menjadi beberapa kelas:

- **Register data**, yang digunakan untuk menyimpan angka-angka dalam bilangan bulat (integer).
- **Register alamat**, yang digunakan untuk menyimpan alamat-alamat memori dan juga untuk mengakses memori.
- **Register *general purpose***, yang dapat digunakan untuk menyimpan angka dan alamat secara sekaligus.
- **Register *floating-point***, yang digunakan untuk menyimpan angka-angka bilangan titik mengambang (floating-point).
- **Register konstanta** (*constant register*), yang digunakan untuk menyimpan angka-angka tetap yang hanya dapat dibaca (bersifat *read-only*), semacam *phi*, *null*, *true*, *false* dan lainnya.
- **Register vektor**, yang digunakan untuk menyimpan hasil pemrosesan vektor yang dilakukan oleh procesor SIMD .
- **Register *special purpose*** yang dapat digunakan untuk menyimpan data internal procesor, seperti halnya instruction pointer, stack pointer, dan status register.
- **Register yang spesifik terhadap model mesin** (*machine-specific register*), dalam beberapa arsitektur tertentu, digunakan untuk menyimpan data atau pengaturan yang berkaitan dengan procesor itu sendiri. Karena arti dari setiap register langsung dimasukkan ke dalam desain procesor tertentu saja, mungkin register jenis ini tidak menjadi standar antara generasi procesor.

2. REGISTER	**PROCESOR**
4-bit	Intel 4004
8-bit	Intel 8080
16-bit	Intel 8086 , Intel 8088 , Intel 80286
32-bit	Intel 80386 , Intel 80486 , Intel Pentium Pro , Intel Pentium , Intel Pentium 2 ,Intel Pentium 3 , Intel Pentium 4 , Intel Celeron , Intel Xeon , AMD K5 , AMD K6 ,AMD Athlon , AMD Athlon MP , AMD Athlon XP , AMD Athlon

	4 , AMD Duron , AMD Sempron
64-bit	Intel Itanium , Intel Itanium 2 , Intel Xeon , Intel Core , Intel Core 2 , AMD Athlon 64 , AMD Athlon X2 , AMD Athlon FX , AMD Turion 64 , AMD Turion X2 , AMD Sempron

Intel 4004 adalah sebuah CPU 4-bit yang merupakan mikroprocesor chip tunggal pertama di dunia. Pada waktu itu, desain CPU lainnya seperti F14 CADC pada tahun 1970 merupakan implementasi dari chip-chip gabungan (*multi-chip*). 4004 dirilis dalam kemasan CERDIP 16-kaki pada tanggal 15 November 1971 . 4004 merupakan procesorkomputer pertama yang dirancang dan diproduksi oleh produsen chip Intel . Orang yang merancang chip tersebut adalah Ted Hoff dan Federico Faggin dari Intel dan Masatoshi Shima dari Busicom.

Rancangan aslinya berasal dari perusahaan Jepang yang bernama Busicom , untuk digunakan pada kalkulator produksinya. 4004 juga disediakan dengan sebuah chip pendukung (misal, ROM program digabung bersama untuk menggunakan alamat program 12-bit 4004, yang mengijinkan akses memori 4 kilobyte dari bus alamat 4-bit bila semua 16 ROM dipasang). Sirkuit 4004 dibuat dari 2.300 transistor , dan pada tahun berikutnya diikuti oleh microprocesor 8-bit pertama, Intel 8008 dengan 3.300 transistor (dan Intel 4040 , perbaikan dari 4004).

Pada masukan ke-empatnya ke pasar mikroprocesor, Intel melepas CPU yang memulai revolusi mikrokomputer ; Intel 8080 .

- Maximum clock speed – nya adalah 740 kHz
- Program dan penyimpanan data yang terpisah (yaitu, sebuah arsitektur Harvard). Berlainan dengan rancangan arsitektur Harvard lainnya yang menggunakan bus yang terpisah, 4004, karena ingin mengurangi jumlah pin, menggunakan sebuah bus 4-bit tunggal dimultiplex untuk mentransfer:
- Alamat 12-bit
- Instruksi 8-bit, tidak ditaruh di memori yang sama dengan
- Data word 4-bit
- Set instruksi yang terdiri dari 46 instruksi (di mana 41 diantaranya memiliki lebar 8 bit dan 5 lebar 16 bit)
- Set register terdiri dari 16 register masing-masing 4 bit
- Tumpukan subroutine internal memiliki kedalaman 3 tingkat
- **Intel Core 2** adalah sebuah mikroprocesor yang dirilis oleh Intel Corporation pada tanggal 27 Juli 2006 . Pada saat pengembangannya, procesor ini memiliki nama kode **Conroe** dan **Allendale**.
- Registers (jamak, dalam bahasa Indonesia menjadi register-register atau banyak register) merupakan media penyimpanan internal CPU yang digunakan saat pengolahan data. Registers merupakan media penyimpanan yang bersifat sementara, artinya data hanya akan berada dalam registers saat data tersebut dibutuhkan selama komputer masih hidup, ketika suatu data tidak diperlukan lagi maka ia tidak berhak lagi berada di dalam registers, dan ketika komputer dimatikan maka semua data yang berada di dalamnya akan hilang.

Instruction cycle (Instruksi Siklus)

Sebuah **siklus instruksi** (kadang disebut **mengambil-dan-execute siklus, mengambil-decode-execute siklus,** atau**FDX)** adalah siklus operasi dasar dari sebuah komputer. Ini adalah proses dimana komputer akan mengambilProgram instruksi dari perusahaan memori , menentukan tindakan apa instruksi membutuhkan, dan melakukan tindakan tersebut. Siklus ini diulang terus menerus oleh unit pengolah pusat (CPU), dari boot untuk saat komputer dimatikan.

Sirkuit digunakan

Sirkuit yang digunakan dalam CPU selama siklus adalah:

- **Program Counter (PC)** – counter incrementing yang melacak alamat memori dari instruksi yang akan dieksekusi selanjutnya ...
- **Memory Address Register (MAR)** – menyimpan alamat dari sebuah blok memori untuk dibaca dari atau ditulis ke
- **Memori data Register (MDR)** – register dua arah yang menyimpan data diambil dari memori (dan siap untuk CPU untuk proses) atau data yang menunggu untuk disimpan dalam memori
- **Instruksi mendaftar (IR)** – tempat memegang sementara untuk instruksi yang baru saja diambil dari memori
- **Control Unit (CU)** – menerjemahkan instruksi program di IR, memilih sumber daya mesin seperti daftar sumber data dan operasi aritmatika tertentu, dan mengkoordinasikan aktivasi sumber daya
- **Aritmatika logika Unit (ALU)** – melakukan operasi matematis dan logis

Periode waktu selama satu instruksi yang diambil dari memori dan dijalankan ketika komputer diberi instruksi dalam bahasa mesin. Ada biasanya empat tahap siklus instruksi bahwa CPU melakukan: 1) Mengambil instruksi dari memori. 2) “Decode” instruksi. 3) “Baca alamat efektif” dari memori jika instruksi memiliki alamat tidak langsung. 4) “Execute” instruksi.

Siklus instruksi CPU Setiap komputer dapat memiliki siklus yang berbeda berdasarkan set instruksi yang berbeda, tetapi akan mirip dengan siklus berikut:

1. Fetch instruksi

Instruksi berikutnya diambil dari alamat memori yang tersimpan saat ini dalam Kontra Program (PC), dan disimpan dalam Instruksi mendaftar (IR). Pada akhir operasi fetch, poin PC ke instruksi berikutnya yang akan dibaca pada siklus berikutnya.

1. Decode instruksi

Decoder menafsirkan instruksi. Selama siklus ini instruksi di dalam IR (instruksi pendaftaran) akan diterjemahkan.

1. In kasus instruksi memori (langsung atau tidak langsung)

Fase eksekusi akan di pulsa clock berikutnya. Jika instruksi memiliki alamat tidak langsung , alamat efektif dibaca dari memori utama, dan setiap data yang dibutuhkan diambil dari memori utama untuk diolah dan kemudian ditempatkan ke dalam register data (Jam Pulse: T 3). Jika instruksi ini langsung, tidak ada yang dilakukan pada pulsa clock. Jika ini adalah instruksi I / O atau instruksi Register, operasi dilakukan (dijalankan) di Pulse jam.

4. Jalankan instruksi

Control Unit CPU melewati informasi dekode sebagai urutan sinyal kontrol ke unit funsi yang relevan dari CPU untuk melakukan tindakan yang dibutuhkan oleh instruksi seperti membaca nilai dari register, melewati mereka ke ALU untuk melakukan fungsi matematika atau logika pada mereka, dan menulis hasilnya kembali ke register. Jika ALU terlibat, ia mengirim sinyal kondisi kembali ke CU tersebut. Hasil yang dihasilkan oleh operasi disimpan dalam memori utama, atau dikirim ke perangkat output. Berdasarkan kondisi umpan balik dari ALU, Counter Program dapat diperbarui ke alamat yang berbeda dari mana instruksi berikutnya akan diambil. Siklus tersebut kemudian diulang.

1. MEMULAI SIKLUS

Siklus dimulai segera pada saat listrik dialirkan ke sistem menggunakan PC nilai awal yang ditetapkan untuk arsitektur sistem (dalam Intel IA-32 CPU, misalnya, nilai PC yang telah ditetapkan adalah 0xfffffff0). Biasanya poin alamat ini dengan instruksi dalam memori hanya-baca (ROM) yang memulai proses loading sistem operasi . (Itu proses loading ini disebut *booting*

2. FETCH SIKLUS

Langkah 1 dari Siklus Instruksi disebut Siklus Fetch. Langkah-langkah ini sama untuk setiap instruksi. Siklus fetch memproses instruksi dari kata instruksi yang berisi opcode .

3. DECODE

Langkah 2 Siklus instruksi disebut membaca sandi tersebut. Opcode diambil dari memori sedang diterjemahkan untuk langkah berikutnya dan pindah ke register yang sesuai.

4. BACA ALAMAT EFEKTIF

Langkah 3 adalah memutuskan yang operasi itu. Jika ini adalah operasi memori – dalam langkah ini komputer memeriksa apakah ini adalah operasi memori langsung atau tidak langsung:

- **Memori instruksi langsung** – Tidak sedang dilakukan.
- **Memori instruksi tidak langsung** – Alamat efektif sedang dibaca dari memori.

Jika ini adalah I / O atau instruksi Daftar – komputer memeriksa jenisnya dan mengeksekusi instruksi.

5. JALANKAN SIKLUS

Langkah 4 dari Siklus Instruksi adalah Siklus Execute. Langkah-langkah ini akan berubah dengan setiap instruksi. Langkah pertama dari siklus eksekusi adalah Proses-Memori. Data ditransfer antara CPU dan modul I / O. Berikutnya adalah Data-Pengolahan menggunakan operasi matematika serta operasi logis dalam referensi data. Perubahan Tengah adalah langkah berikutnya, adalah urutan operasi, misalnya operasi melompat. Langkah terakhir adalah operasi gabungan dari semua langkah lainnya.

6. SIKLUS FETCH-EXECUTE DALAM NOTASI TRANSFER

Register yang digunakan di atas, selain yang dijelaskan sebelumnya, yang Address Memori Register (MAR) dan Memory Data Register (MDR), yang digunakan (setidaknya secara

konseptual) dalam mengakses memori. Ambil dan mengeksekusi contoh (ditulis dalam RTL – Register Bahasa Transfer):

PC = 0x5AF, AC = 0x7EC3, M [0x5AF] = 0x932E, M [0x32E] = 0x09AC, M [0x9AC] = 0x8B9F.

T0: AR = 0x5AF (PC)
T1: IR = 0x932E (M [AR]), PC = 0x5BO
T2: deCODE = 0x932E opcode ADD, AR = 0x32E, I = 1. (Instruksi langsung)
T3: AR = 0x9AC (M [AR])
T4: DR = 0x8B9F
T5: AC = 0x8B9F + 0x7EC3 = 0x0A62, E = 1 (melaksanakan), SC = 0

Ringkasan: contoh ini adalah untuk Instruksi ADD yang dibuat tidak langsung dimana:
T0-T1 adalah operasi Ambil.
T2 adalah Decode kode operasi.
T3 Memori referensi tidak langsung
T4-T5 Execute ADD operasi

Data flow diagram

Data Flow Diagram (DFD) adalah suatu diagram yang menggunakan notasi-notasi untuk menggambarkan arus dari data sistem, yang penggunaannya sangat membantu untuk memahami sistem secara logika, tersruktur dan jelas.

DFD merupakan alat bantu dalam menggambarkan atau menjelaskan sistem yang sedang berjalan logis. Suatu yang lazim bahwa ketika menggambarkan sebuah sistem kontekstual data flow diagram yang akan pertama kali muncul adalah interaksi antara sistem dan entitas luar. DFD didisain untuk menunjukkan sebuah sistem yang terbagi-bagi menjadi suatu bagian sub-sistem yang lebih kecil adan untuk menggarisbawahi arus data antara kedua hal yang tersebut diatas. Diagram ini lalu "dikembangkan" untuk melihat lebih rinci sehingga dapat terlihat model-model yang terdapat di dalamnya.

Condition code register

CCR atau condition code register adalah register dengan ukuran 8 bit, yang menyimpan indikator status dari hasil operasi CPU sebelumnya. Terdapat tiga bit teratas tidak digunakan dan selalu sama dengan logika satu. Instruksi percabangan menggunakan bit-bit status dalam register ini untuk mengerjakan suatu keputusan percabangan. Isi dari CCR ini adalah sebuah interrupt mask dan empat indikator status. Kelima flag tersebut adalah H atau half cary, N atau negative, Z atau zero, dan C atau carry/borrow.

Flag H (half carry) digunakan untuk operasi aritmatika BCD (Binary Coded Decimal) dan dipengaruhi oleh kerja instruksi ADD dan ADC. Bit H ini akan set jika ada carry yang timbul dari digit heksadesimal 0-3 (low order) dan digit desimal 4-7 (high order).

Bit I (interrupt mask) bukanlah status flag tetapi merupakan bit yang akan men-nonaktif-kan semua sumber interrupt yang maskable saat bit ini diset. Interrupt baru akan aktif jika bit ini nol. Jika ada interrupt eksternal yang terjadi saat bit I diset, maka interrupt tersebut akan di-latch dan akan diproses saat bit I dinolkan. Karena itu, interrupt yang terjadi tidak akan hilang. Setelah interrupt ditangani, instruksi RTI (return from interrupt) akan menyebabkan register ini dikembalikan ke nilai semula. Umumnya, bit I ini akan menjadi nol setelah instruksi RTI dilaksanakan.

Flag N (negative) akan diset jika hasil dari operasi aritmatika, logika, maupun manipulasi data yang terakhir adalah negatif. Nilai negatif dalam two's complement ditandai jika bit MSB adalah satu.

Flag Z (zero) diset jika hasil dari operasi aritmatika, logika, maupun manipulasi data terakhir adalah nol. Instruksi perbandingan (compare) akan mengurangi suatu harga dari suatu lokasi memori yang akan dites. Jika nilainya sama, maka bit Z ini akan diset.

Flag C (carry/borrow) digunakan untuk menandai apakah ada carry dari hasil operasi tambah atau ada borrow darioperasi pengurangan. Instruksi shift dan rotate juga dapat memakai bit C ini.

SP atau stack pointer digunakan sebagai pointer ke lokasi yang tersedia berikutnya dalam tumpukan stack dalam uruatn LIFO (last-in first-out). Stack ini dapat dianalogikan sebagai tumpukan kartu. Setiap kartu menyimpan satu byte (8 bit) informasi. Dalam suatu saat, CPU dapat menaruh satu kartu di atas tumpukan kartu tersebut maupun mengambil satu kartu dari tumpukan. Kartu di dalam tumpukan tidak dapat diambil kecuali jika kartu di atasnya sudah diambil sebelumnya. Jika CPU menaruh informasi dalam stack, maka data tersebut akan dituliskan dalam memori yang ditunjukkan oleh nilai SP saat itu, dan kemudian nilai SP akan dikurangi satu sehingga SP akan menunjukkan ke lokasi memori berikutnya yang kosong untuk digunakan sebagai penyimpan berikutnya. Jika CPU mengambil data dari stack, SP akan ditambah satu sehingga menunjukkan ke lokasi stack yang terakhir, dan kemudian data diambil dan dibaca oleh CPU. Saat CPU pertama kali dihidupkan atau setelah instruksi Reset Stack Pointer (RSP), maka SP akan menunjukkan memori tertentu RAM.

PIPELINE

Lima tahap pipeline pada mesin RISC (IF = Instruction Fetch(FI), ID = Instruction Decode(DI), EX = Execute(EI), MEM = Memory access, WB = Register write back(WO)) Dalam siklus waktu keempat (kolom hijau), paling awal adalah instruksi dalam tahap Mem, dan instruksi terbaru belum masuk set pipelining.

Gambar pipeline :

Pipeline adalah suatu cara yang digunakan untuk melakukan sejumlah kerja secara bersama tetapi dalam tahap yang berbeda yang dialirkan secara kontinu pada unit pemrosesor. Dengan cara ini, maka unit pemrosesan selalu bekerja. Teknik pipeline ini dapat diterapkan pada berbagai tingkatan dalam sistem komputer. Bisa pada level yang tinggi, misalnya program aplikasi, sampai pada tingkat yang rendah, seperti pada instruksi yang dijaankan oleh*microprocessor*.
Pada *microprocessor* yang tidak menggunakan pipeline, satu instruksi dilakukan sampai selesai, baru instruksi berikutnya dapat dilaksanakan. Sedangkan dalam *microprocessor* yang menggunakan teknik pipeline, ketika satu instruksi sedangkan diproses, maka instruksi yang berikutnya juga dapat diproses dalam waktu yang bersamaan. Tetapi, instruksi yang diproses secara bersamaan ini, ada dalam tahap proses yang berbeda. Jadi, ada sejumlah tahapan yang akan dilewati oleh sebuah instruksi.

Dengan penerapan pipeline ini pada *microprocessor* akan didapatkan peningkatan kinerja *microprocessor*. Hal ini terjadi karena beberapa instruksi dapat dilakukan secara parallel dalam waktu yang bersamaan. Secara kasarnya diharapkan akan didapatkan

peningkatan sebesar K kali dibandingkan dengan *microprocessor* yang tidak menggunakan pipeline, apabila tahapan yang ada dalam satu kali pemrosesan instruksi adalah *K* tahap.

Karena beberapa instruksi diproses secara bersamaan ada kemungkinan instruksi tersebut sama-sama memerlukan resource yang sama, sehingga diperlukan adanya pengaturan yang tepat agar proses tetap berjalan dengan benar dan lancar. Sedangkan ketergantungan terhadap data bisa muncul, misalnya instruksi yang berurutan memerlukan data dari instruksi yang sebelumnya. Kasus Jump, juga perlu perhatian, karena ketika sebuah instruksi meminta untuk melompat ke suatu lokasi memori tertentu, akan terjadi perubahan program counter, sedangkan instruksi yang sedang berada dalam salah satu tahap proses yang berikutnya mungkin tidak mengharapkan terjadinya perubahan program counter.

Teknik pipeline yang diterapkan pada *microprocessor*, dapat dikatakan sebuah arsitektur khusus. Ada perbedaan khusus antara model *microprocessor* yang tidak menggunakan arsitektur pipeline dengan *microprocessor* yang menerapkan teknik ini.

Pada *microprocessor* yang tidak menggunakan pipeline, satu instruksi dilakukan sampai selesai, baru instruksi berikutnya dapat dilaksanakan. Sedangkan dalam *microprocessor* yang menggunakan teknik pipeline, ketika satu instruksi sedangkan diproses, maka instruksi yang berikutnya juga dapat diproses dalam waktu yang bersamaan. Tetapi, instruksi yang diproses secara bersamaan ini, ada dalam tahap proses yang berbeda.

Jadi, ada sejumlah tahapan yang akan dilewati oleh sebuah instruksi. Misalnya sebuah*microprocessor* menyelesaikan sebuah instruksi dalam 4 langkah. Ketika instruksi pertama masuk ke langkah 2, maka instruksi berikutnya diambil untuk diproses pada langkah 1 instruksi tersebut. Begitu pun seterusnya, ketika instruksi pertama masuk ke langkah 3, instruksi kedua masuk ke langkah 2 dan instruksi ketiga masuk ke langkah 1.

Teknik pipeline ini menyebabkan ada sejumlah hal yang harus diperhatikan sehingga ketika diterapkan dapat berjalan dengan baik.
Tiga kesulitan yang sering dihadapi ketika menggunakan teknik pipeline ini adalah :

1. Terjadinya penggunaan resource yang bersamaan
2. Ketergantungan terhadap data, dan
3. Pengaturan Jump ke suatu lokasi memori.

1. INSTRUKSI PADA PIPELINE

Tahapan pipeline

- Mengambil instruksi dan membuffferkannya
- Ketika tahapan kedua bebas tahapan pertama mengirimkan instruksi yang dibufferkan tersebut
- Pada saat tahapan kedua sedang mengeksekusi instruksi ,tahapan pertama memanfaatkan siklus memori yang tidak dipakai untuk mengambil dan membuffferkan instruksi berikutnya

Berikut ini adalah gambaran tentang Instuksi pipeline :

Karena untuk setiap tahap pengerjaan instruksi, komponen yang bekerja berbeda, maka dimungkinkan untuk mengisi kekosongan kerja di komponen tersebut. Sebagai contoh

Instruksi 1 : ADD AX, AX Instruksi 2: ADD EX,CX

Setelah CU menjemput instruksi 1 dari memori (IF), CU akan menerjemahkan instruksi tersebut(ID). Pada menerjemahkan instruksi 1 tersebut, komponen IF tidak bekerja. Adanya teknologi pipeline menyebabkan IF akan menjemput instruksi 2 pada saat ID menerjemahkan instruksi 1. Demikian seterusnya pada saat CU menjalankan instruksi 1 (EX), instruksi 2 diterjemahkan (ID).

Contoh pengerjaan instruksi tanpa pipeline :

Contoh pengerjaan instruksi dengan pipeline :

Dengan adanya pipeline dua instruksi selesai dilaksanakan padadetik keenam (sedangkan pada kasus tanpa pipeline baru selesai pada detik kesepuluh). Dengan demikian telah terjadi percepatan sebanyak 1,67x dari 10T menjadi hanya 6T. Sedangkan untuk pengerjaan 3 buah instruksi terjadi percepatan sebanyak 2, 14x dari 15T menjadi hanya 7T.

Untuk kasus pipeline sendiri, 2 instruksi dapat dikerjakan dalam 6T (CPI = 3) dan instruksi dapat dikerjakan dalam 7T (CPT = 2,3) dan untuk 4 instruksi dapat dikerjakan dalam 8T (CPI =2). Ini berarti untuk 100 instruksi akan dapat dikerjakan dalam 104T (CPI = 1,04). Pada kondisi ideal CPI akan harga 1**.**

2. KONSEP PIPELINE

Konsep pemrosesan pipeline dalam suatu komputer mirip dengan suatu baris perakitan dalam suatu pabrik industri. Ambil contoh, suatu proses pembuatan sebuah mobil: anggaplah bahwa langkah-langkah tertentu di jalur perakitan adalah untuk memasang mesin, memasang kap mesin, dan memasang roda (dalam urutan tersebut, dengan langkah arbitrary interstitial). Sebuah mobil di jalur perakitan hanya dapat memiliki salah satu dari tiga tahap yang dilakukan sekaligus.

Setelah mobil memiliki mesin yang terpasang, bergerak ke bagian pemasangan kap, meninggalkan fasilitas pemasangan mesin yang tersedia untuk mobil berikutnya. Mobil pertama kemudian pindah ke pemasangan roda, mobil kedua untuk pemasangan kap, dan mobil ketiga dimulai untuk pemasangan mesin. Jika instalasi mesin membutuhkan waktu 20 menit, instalasi kap mobil memakan waktu 5 menit, dan instalasi roda membutuhkan waktu 10 menit, kemudian menyelesaikan semua tiga mobil ketika hanya satu mobil dapat dioperasikan sekaligus akan memakan waktu 105 menit.

Di sisi lain, dengan menggunakan jalur perakitan, total waktu untuk menyelesaikan ketiga adalah 75 menit. Pada titik ini, mobil selanjutnya akan datang dari jalur perakitan pada kenaikan 20 menit.

3. MASALAH-MASALAH PADA PIPELINE

Dengan adanya persyaratan bahwa setiap instuksi yang berdekatan harus tidak saling bergantung, maka ada kemungkinan terjadinya situasi dimana pipeline gagal dilaksanakan (instruksi berikutnya tidak bisa dilaksanakan). Situasi ini disebut *Hazards*. *Hazards* mengurangi performansi dari *CPU* dimana percepatan ideal tidak dapat dicapai.

Ada 3 kelompok ***Hazards*** :

1. ***Structural Hazards*** muncul dari konflik resource sistem yaitu ketika hardware tidak dapat mensuport semua kemungkinan kombinasi pelaksanaan instruksi.
2. ***Data Hazards*** muncul ketika data untuk suatu instruksi tergantung pada hasil instruksi sebelumnya.
3. ***Control Hazards*** muncul pada pelaksanaan instruksi yang mengubah PC (*contoh* : branch).

Adanya *Hazards* menyebabkan pipeline terhambat (*stalled*). Tidak ada instruksi baru yang dijemput sampai hambatan itu selesai. Ini berarti instruksi-instruksi selanjutnya akan ditunda pula penjemputannya**.**

4. KEUNTUNGAN DARI PIPELINE

1. Waktu siklus procesor berkurang, sehingga meningkatkan tingkat instruksi-isu dalam kebanyakan kasus.
2. Beberapa combinational sirkuit seperti penambah atau pengganda dapat dibuat lebih cepat dengan menambahkan lebih banyak sirkuit.

Jika pipeline digunakan sebagai pengganti, hal itu dapat menghemat sirkuit vs combinational yang lebih kompleks sirkuit.

5. KERUGIAN DARI PIPELINE

1. Prossesor non-pipeline hanya menjalankan satu instruksi pada satu waktu. Hal ini untuk mencegah penundaan cabang (yang berlaku, setiap cabang tertunda) dan masalah dengan serial instruksi dieksekusi secara bersamaan. Akibatnya desain lebih sederhana dan lebih murah untuk diproduksi.
2. Instruksi latency di prossesor non-pipeline sedikit lebih rendah daripada dalam pipeline setara. Hal ini disebabkan oleh fakta bahwa sandal jepit ekstra harus ditambahkan ke jalur data dari prossesor pipeline.
3. Prossesor non-pipeline akan memiliki instruksi bandwidth yang stabil. Kinerja prossesor yang pipeline jauh lebih sulit untuk meramalkan dan dapat bervariasi lebih luas di antara program yang berbeda.

Pipelining, merupakan fitur standar pada procesor RISC, tidak sama dengan assembly line. Karena procesor bekerja pada berbagai langkah dari instruksi pada saat yang sama, beberapa pekerjaan bisa dilaksanakan dalam jangka waktu yang lebih singkat. Teknologi Pipeline yang digunakan pada komputer bertujuan untuk meningkatkan kinerja dari komputer.

Teknik pipeline yang diterapkan pada microprocessor, dapat dikatakan sebuah arsitektur khusus. Ada perbedaan khusus antara model microprocessor yang tidak menggunakan arsitektur pipeline dengan microprocessor yang menerapkan teknik ini.

1. Pada microprocessor yang tidak menggunakan pipeline, satu instruksi dilakukan sampai selesai, baru instruksi berikutnya dapat dilaksanakan.
2. Pada microprocessor yang menggunakan teknik pipeline, ketika satu instruksi sedangkan diproses, maka instruksi yang berikutnya juga dapat diproses dalam waktu yang bersamaan. Tetapi, instruksi yang diproses secara bersamaan ini, ada dalam tahap proses yang berbeda. Jadi, ada sejumlah tahapan yang akan dilewati oleh sebuah instruksi.

Misalnya sebuah microprocessor menyelesaikan sebuah instruksi dalam 4 langkah, Ketika instruksi pertama masuk ke langkah 2, maka instruksi berikutnya diambil untuk diproses

pada langkah 1 instruksi tersebut. Begitu seterusnya, ketika instruksi pertama masuk ke langkah 3, instruksi kedua masuk ke langkah 2 dan instruksi ketiga masuk ke langkah 1. Dengan penerapan pipeline ini pada microprocessor akan didapatkan peningkatan dalam unjuk kerja microprocessor.

Hal ini terjadi karena beberapa instruksi dapat dilakukan secara parallel dalam waktu yang bersamaan. Secara kasarnya diharapkan akan didapatkan peningkatan beberapa kali dibandingkan dengan microprocessor yang tidak menggunakan pipeline, apabila tahapan yang ada dalam satu kali pemrosesan instruksi adalah banyak tahap.
Teknik pipeline ini menyebabkan ada sejumlah hal yang harus diperhatikan sehingga ketika diterapkan dapat berjalan dengan baik.

Tiga kesulitan yang sering dihadapi ketika menggunakan teknik pipeline ini adalah :

1. Terjadinya penggunaan resource yang bersamaan,
2. Ketergantungan terhadap data, dan
3. Pengaturan Jump ke suatu lokasi memori.

Karena beberapa instruksi diproses secara bersamaan ada kemungkinan instruksi tersebut sama-sama memerlukan resource yang sama, sehingga diperlukan adanya pengaturan yang tepat agar proses tetap berjalan dengan benar. Sedangkan ketergantungan terhadap data, bisa muncul, misalnya instruksi yang berurutan memerlukan data dari instruksi yang sebelumnya. Kasus Jump, juga perlu perhatian, karena ketika sebuah instruksi meminta untuk melompat ke suatu lokasi memori tertentu, akan terjadi perubahan program counter, sedangkan instruksi yang sedang berada dalam salah satu tahap proses yang berikutnya mungkin tidak mengharapkan terjadinya perubahan program counter.

Dengan menerapkan teknik pipeline ini, akan ditemukan sejumlah perhatian yang khusus terhadap beberapa hal di atas, tetapi tetap akan menghasilkan peningkatan yang berarti dalam kinerja microprocessor. Ada kasus tertentu yang memang sangat tepat bila memanfaatkan pipeline ini, dan juga ada kasus lain yang mungkin tidak tepat bila menggunakan teknologi pipeline.

6. GENERIC PIPELINE

Ada 4 tahapan dalam generic pipeline :
1. Fetch
2. Decode
3. Execute
4. Write-back

PERFORMA PIPELINE

Procesor pipelined menyelesaikan pengolahan satu instruksi pada tiap cycle, yang berarti kecepatan pengolahan instruksi tersebut empat kali lebih besar dari op e. berurutan. Peningkatan potensial dalam performa yang dihasilkan dari pipelining proporj,_M dengan jumlah pipeline stage. Akan tetapi, peningkatan ini hanya akan dicapai jika ope pipelined dapat dipertahankan tanpa inter melalui eksekusi program. Stalled selama dua clock cycle. pipelined normal dimulai lagi pada cycle 7. Tiap kondisi yang menyebabkan pipeline stall but hazard.

https://retnoree.wordpress.com/2012/06/07/makalah-struktur-cpu/