Komputer telah menjadi bagian dalam kehidupan sehari-hari. Secara umum computer dapat dibagi menjadi tiga kategori, yaitu PC, Server, dan Embedded Computer. PC adalah komputer personal yang sehari-hari dipakai oleh manusia dalam bentuk *laptop* dan *desktop*. PC pada umumnya memiliki kemampuan untuk mengolah multimedia, memiliki harga terjangkau, dan multifungsi. Server biasanya ditemukan pada *data center* dan berfungsi untuk mengolah data berukuran besar atau melakukan komputasi yang tidak mampu dilakukan oleh PC. Pada dasarnya PC mampu melakukan komputasi tersebut tetapi PC memerlukan waktu yang sangat lama jika dibandingkan dengan server. Selain itu, server memiliki *cost* yang tinggi baik dari sisi harga komponen hardware, software, maupun biaya operasional. Embedded Computer adalah komputer yang tertanam pada hardware. Karakteristik dari komputer jenis ini adalah memiliki fitur yang terbatas sehingga pada umumnya tidak mendukung instalasi *third party software*. Komputer jenis ini lebih murah dari PC atau server.

Komputer memiliki empat komponen utama, yaitu CPU atau unit pemrosesan utama, memori, perangkat input, dan perangkat output. CPU berfungsi untuk menjalankan instruksi-instruksi yang berasal dari program. Memori adalah perangkat yang menyimpan data dan instruksi yang sedang diproses oleh CPU. Perangkat input dan output adalah perangkat yang digunakan untuk menginput data dari luar komputer untuk diproses oleh CPU dan menampilkan hasil pemrosesan tersebut.

CPU mengeksekusi program dalam bentuk mengeksekusi instruksi-instruksi yang tertera pada program. Pada prinsipnya program di-*compile* menjadi kumpulan instruksi untuk dieksekusi oleh CPU. Agar berbagai program dalam dijalankan pada CPU, perlu didefinisikan kumpulan instruksi yang dapat dieksekusi CPU. Instruksi-instruksi tersebut meliputi operasi aritmatika, transfer data, logika, *conditional branch*, dan *unconditional branch*.

ISA atau Instruction Set Architecture adalah set instruksi yang menjadi jembatan antara software dan hardware. ISA berfungsi untuk mendefinikasi instruksi dan *bitwise operation* pada hardware. Hal ini bertujuan agar memudahkan berbagai software dapat diimplementasi pada hardware yang berbeda selama hardware tersebut menggunakan ISA yang sama. ISA pada umumnya terbagi menjadi dua jenis, yaitu RISC (*reduced instruction set computer*) dan CISC (*complex instruction set computer*). Perbedaan mendasar pada kedua jenis ISA ini adalah RISC memilih untuk menyederhanakan instruksi-instruksi sehingga ukuran instruksi pendek. Instruksi yang pendek memungkinkan instruksi tersebut dibuat dalam bentuk hardware sehingga waktu eksekusinya lebih cepat. ISA yang dibahas dalam ringkasan ini adalah MIPS (microprocessor without interlocked pipeline stages) yang termasuk dalam RISC. Instruksi MIPS memiliki tiga jenis instruksi dengan panjang 32 bit, yaitu R-Type *instruction* atau instruksi aritmatika, *I-Type instruction* atau *branching instruction*, dan *J-Type instruction* atau instruksi *unconditional branching*.

Performa CPU dihitung dari response time dan throughput. Response time adalah waktu yang dibutuhkan komputer untuk merespon input dari user. Throughput adalah ukuran seberapa banyak instruksi yang dapat dieksekusi oleh komputer dalam satu waktu. Kedua hal ini berkaitan dengan execution time, yaitu waktu yang dibutuhkan CPU untuk mengeksekusi program user.

Untuk meningkatkan kemampuan CPU untuk mengeksekusi program user, maka CPU dikembangkan untuk dapat melakukan proses *pipelining*. CPU memiliki berbagai unit kerja. Beberapa diantaranya unit aritmatika, unit *branch control*, unit *load* untuk mengambil instruksi dan data dari memori, dan unit *store* untuk menyimpan data hasil eksekusi. Satu instruksi tidak menggunakan semua unit kerja pada satu waktu sehingga ketika suatu instruksi dieksekusi, selalu ada unit kerja yang tidak digunakan.

Untuk memaksimalkan penggunakan CPU, suatu instruksi dibagi menjadi beberapa fase atau *stage*. Fase pertama adalah membaca instruksi dari memori. Fase kedua adalah membaca register. Fase ketiga adalah eksekusi instruksi. Fase keempat adalah membaca operan pada memori. Fase kelima adalah menulis hasil eksekusi pada register. Dengan demikian, jika instruksi pertama telah selesai dibaca dari memori, CPU dapat langsung membaca instruksi kedua tanpa harus menunggu instruksi pertama selesai dieksekusi. Hal ini berlaku pada instruksi-instruksi selanjutnya.

Pada prakteknya, *pipeline* tidak selalu berjalan dengan dengan baik. Ada beberapa hambatan atau *hazard* dalam praktek *pipelining* instruksi yang membuat eksekusi instruksi-instruksi pada *pipeline* harus ditunda atau *stalled*. *Hazard* ini secara umum terbagi menjadi tiga jenis, yaitu *structural hazard*, *control hazard* dan *data hazard*. Structural hazard berkaitan dengan kemampuan hardware untuk mengeksekusi instruksi dalam *pipeline*.

Control Hazard berkaitan dengan kondisi *branching*. Jika terdapat instruksi *branching*, maka instruksi yang terkait dengan kondisi *branching* tersebut tidak boleh dieksekusi sebelum kondisi *branching* diketahui. Data Hazard berkaitan dengan ketergantungan antar data.

*Data hazard* terbagi menjadi dua, yaitu *true dependence* dan *name dependence*. True dependence terjadi Ketika suatu instruksi *j* akan membaca suatu register *r* yang merupakan tempat menyimpan hasil dari instruksi sebelumnya, misalkan, instruksi *i*. Urutan eksekusi kedua instruksi ini harus dijaga agar hasil eksekusi tetap konsisten. *Name dependence* terjadi Ketika dua instruksi yang berbeda akan menulis ke register yang sama. Hal ini dapat ditangani dengan cara mengganti register yang sama dengan register yang baru.

Selain menggunakan pipeline, cara lain yang dapat digunakan untuk meningkatkan parallelism sehingga mempercepat waktu eksekusi program adalah dengan menggunakan multiprosesor. Untuk dapat memaksimalkan penggunaan multiprosesor, *task* atau program harus dipecah menjadi beberapa bagian yang independent untuk dieksekusi pada multiprosesor. Pembagian tugas ini penting untuk menjadi masing-masing CPU dapat mengeksekusi instruksi tanpa harus bergantung dengan eksekusi CPU lain.

Salah satu cara mengeksekusi program secara paralel dengan menggunakan prosesor berjenis SIMD, yaitu single instruction multi data stream. Prosesor ini mengolah data dalam bentuk vector sehingga dalam satu fase, instruksi dapat mengolah data lebih dari satu. Cara lain untuk meningkatkan kecepatan eksekusi program adalah dengan memecah *task* menjadi beberapa bagian untuk dieksekusi oleh beberapa CPU secara paralel. Dengan kata lain, berbagai *data stream* dapat dieksekusi oleh beberapa aliran instruksi. Sehingga, pada arsitektur multiprosesor digunakan prosesor berjenis *multi instruction stream multi data stream* atau MIMD.

Instruksi dan data yang digunakan pada proses eksekusi disimpan pada memori. Pada eksekusi instruksi, kecepatan memori menjadi krusial untuk dipertimbangkan untuk menjamin CPU selalu mendapatkan instruksi dan data untuk dieksekusi. Jika kecepatan memori lebih rendah dari CPU maka CPU akan mengalami waktu *idle* sehingga utilisasi CPU tidak dapat maksimal. Memori memiliki kecepatan dan kapasitas yang berbeda-beda. Semakin dekat posisi memori dengan CPU, memori tersebut semakin cepat dan kapasitasnya semakin kecil. Sebaliknya, semakin jauh letak memori dari CPU, semakin lambat kecepatan memori tersebut dan semakin besar kapasitasnya. Register adalah memori yang berada di dalam *chip* CPU. Pada arsitektur MIPS, register yang tersedia berjumlah 32 register. Bentuk memori adalah RAM atau random access memory, flash disk, dan hard disk.