**Organisasi dan Arsitektur Komputer**  
**Set Instruksi Intel x86**

**x86**

**x86** atau **80x86** adalah nama umum dari arsitektur mikroprosesor yang pertama kali dikembangkan dan diproduksi oleh Intel. Arsitektur **x86** saat ini mendominasi komputer desktop, komputer portabel, dan pasar server sederhana.

**Keping Mikroposesor Intel Pentium 4; Seri Northwood**

Arsitektur ini dikenal dengan nama x86 karena prosesor-prosesor awal dari keluarga arsitektur ini memiliki nomor model yang diakhiri dengan urutan angka "86": prosesor 8086, 80186, 80286, 386, dan 486. Karena nomor tidak bisa dijadikan merek dagang, Intel akhirnya menggunakan kata Pentium untuk merek dagang processor generasi kelima mereka. Arsitektur ini telah dua kali diperluas untuk mengakomodasi ukuran word yang lebih besar. Pada tahun 1985, Intel mengumumkan rancangan generasi 386 32-bit yang menggantikan rancangan generasi 286 16-bit. Arsitektur 32-bit ini dikenal dengan nama x86-32 atau IA-32 (singkatan dari Intel Architecture, 32-bit). Kemudian pada tahun 2003, AMD memperkenalkan Athlon 64, yang menerapkan secara lebih jauh pengembangan dari arsitektur ini menuju ke arsitektur 64-bit, dikenal dengan beberapa istilah x86-64, AMD64 (AMD), EM64T atau IA-32e (Intel), dan x64 (Microsoft).

**Jenis x86 data**

x86 dapat menangani jenis data 8 (byte), 16 (kata), 32 (doubleword), 64 (quadword), dan 128 (double quadword) bit panjangnya. Untuk memungkinkan fleksibilitas maksimum dalam struktur data dan pemanfaatan memori efisien, kata tidak perlu selaras pada evennumbered alamat; doublewords tidak perlu selaras pada alamat merata dibagi jadi 4; dan quadwords tidak perlu selaras pada alamat dibagi secara merata 8; dan sebagainya.

**Rancangan**

Arsitektur x86 adalah rancangan Set Instruksi Komputer Kompleks (Complex Instruction Set Computer) dengan panjang instruksi yang bervariasi. Word disimpan dengan urutan endian-kecil. Kompatibilitas mundur menjadi motivasi terkuat dalam pengembangan arsitektur x86 (keputusan ini menjadi sangat penting dan sering dikritik, terutama oleh pesaing dari pendukung arsitektur prosesor lainnya, yang dibuat frustasi oleh sukses yang berkelanjutan dari arsitektur ini yang secara umum dipandang memilki banyak kelemahan). Prosesor-prosesor terkini dari x86 menerapkan beberapa langkah penerjemah (dekoder) "tambahan" untuk (saat eksekusi) memecah (sebagian besar) instruksi x86 kedalam potongan-potongan kecil instruksi (dikenal dengan "micro-ops") yang selanjutnya dieksekusi oleh arsitektur setara dengan arsitektur RISC.

Mikroprosesor x86 dapat bekerja dalam beberapa modus berikut:

* [Real-mode](https://github.com/L200150043/Organisasi-dan-Arsitektur-Komputer/blob/master/Tugas-01%20-%20Set%20instruksi%20x86.md#real-mode) (Modus Real)
* [Protected Mode](https://github.com/L200150043/Organisasi-dan-Arsitektur-Komputer/blob/master/Tugas-01%20-%20Set%20instruksi%20x86.md#protected-mode) (Modus terproteksi)
* [Virtual Protected Mode](https://github.com/L200150043/Organisasi-dan-Arsitektur-Komputer/blob/master/Tugas-01%20-%20Set%20instruksi%20x86.md#virtual-protected-mode) (Modus Terproteksi Virtual)
* [Compatibility Mode](https://github.com/L200150043/Organisasi-dan-Arsitektur-Komputer/blob/master/Tugas-01%20-%20Set%20instruksi%20x86.md#ia32eamd64x64x86-64-compatibility-mode)
* [Long Mode/IA32e Full Mode](https://github.com/L200150043/Organisasi-dan-Arsitektur-Komputer/blob/master/Tugas-01%20-%20Set%20instruksi%20x86.md#ia32ex86-64amd64x64em64t-long-mode)

**Real-Mode**

Real-Mode adalah sebuah modus di mana prosesor Intel x86 berjalan seolah-olah dirinya adalah sebuah prosesor Intel 8086 atau Intel 8088, meski ia merupakan prosesor Intel 80286 atau lebih tinggi. Karenanya, modus ini juga disebut sebagai modus 8086 (8086 Mode). Dalam modus ini, prosesor hanya dapat mengeksekusi instruksi 16-bit saja dengan menggunakan register internal yang berukuran 16-bit, serta hanya dapat mengakses hanya 1024 KB dari memori karena hanya menggunakan 20-bit jalur bus alamat. Semua program DOS berjalan pada modus ini.

Prosesor yang dirilis setelah 8086, semacam Intel 80286 juga dapat menjalankan instruksi 16-bit, tapi jauh lebih cepat dibandingkan 8086. Dengan kata lain, Intel 80286 benar-benar kompatibel dengan prosesor Intel 8086 yang didesain sebelumnya. Sehingga prosesor Intel 80286 pun dapat menjalankan program-program 16-bit yang didesain untuk 8086 (IBM PC), dengan tentunya kecepatan yang jauh lebih tinggi.

Dalam Real-mode, tidak ada proteksi ruang alamat memori, sehingga tidak dapat melakukan multi-tasking. Inilah sebabnya, mengapa program-program DOS bersifat single-tasking. Jika dalam modus real terdapat multi-tasking, maka kemungkinan besar antara dua program yang sedang berjalan, terjadi tabrakan (crash) antara satu dengan lainnya.

**Protected Mode**

Modus terproteksi (protected mode) adalah sebuah modus di mana terdapat proteksi ruang alamat memori yang ditawarkan oleh mikroprosesor untuk digunakan oleh sistem operasi. Modus ini datang dengan mikroprosesorIntel 80286 atau yang lebih tinggi. Karena memiliki proteksi ruang alamat memori, maka dalam modus ini sistem operasi dapat melakukan multitasking.

Prosesor Intel 80286 memang dilengkapi kemampuan masuk ke dalam modus terproteksi, tapi tidak dapat keluar dari modus tersebut tanpa harus mengalami reset (warm boot atau cold boot). Kesalahan ini telah diperbaiki oleh Intel dengan merilis prosesor Intel 80386 yang dapat masuk ke dalam modus terproteksi dan keluar darinya tanpa harus melakukan reset. Inilah sebabnya mengapa Windows 95/Windows 98 dilengkapi dengan modus Restart in MS-DOS Mode, meski sebenarnya sistem operasi tersebut merupakan sistem operasi yang berjalan dalam modus terproteksi.

**Virtual Protected Mode**

Virtual Protected Mode juga kadang disebut sebagai Virtual Real Mode. Dalam modus ini, sebuah prosesor Intel x86 berjalan dalam modus terproteksi tetapi mengizinkan aplikasi-aplikasi 16-bit real-mode agar dapat dijalankan di atas sistem operasi. Microsoft Windows 3.1, yang berjalan di dalam modus Enhanced 386, Windows 95, serta Windows 98 mendukung modus ini sepenuhnya. Sistem-sistem operasi dapat menjalankan beberapa aplikasi 16-bit real-mode secara sekaligus, pada window MS-DOS Prompt yang berbeda-beda, karena memang Microsoft mengimplementasikan sebuah lapisan emulasi yang disebut sebagai DOS Protected Mode Interface (DPMI). Setiap window MS-DOS Prompt yang dibuat, aplikasi hanya berjalan dalam real mode, tapi karena Windows 3.1 (yang berjalan dalam modus Enhanced 386) dan Windows 95/98 berjalan dalam modus terproteksi, aplikasi akan menganggap dirinya berjalan pada komputer yang berbeda, meski pada fisiknya mereka dijalankan pada modus yang sama. Hal ini mengizinkan aplikasi-aplikasi 16-bit real-mode agar dapat dijalankan secara serentak (multitasking), meski pada awalnya aplikasi 16-bit berjalan dalam kondisi single-tasking.

DPMI digunakan oleh Windows 3.1 ke atas untuk mengakses extended memory agar dapat digunakan oleh aplikasi Windows. DPMI mengizinkan program-program dapat menggunakan memori yang lebih banyak, meski pada aslinya program tersebut merupakan program 16-bit. Hal ini populer dilakukan, khususnya bagi program-program game komputer DOS, karena game-game tersebut dapat mengakses lebih dari 1 MB (diberi hak akses oleh sistem operasi). DPMI dapat melakukan switching prosesor dari real-mode ke protected mode atau sebaliknya.

**IA32e/AMD64/x64/x86-64 Compatibility Mode**

Modus kompatibilitas adalah sebuah modus prosesor berbasis IA32e (x86-64, AMD64, EM64T, atau x64) di mana prosesor sedang menjalankan instruksi 32-bit (sistem operasi 32-bit dan aplikasinya yang dijalankan di atas prosesor x64 atau sistem operasi 64-bit yang menjalankan aplikasi 32-bit). Dalam modus ini, prosesor tersebut bekerja seolah-olah dirinya adalah prosesor x86 32-bit, sehingga hanya dapat mengalamati memori hingga 4 GB saja.

**IA32e/x86-64/AMD64/x64/EM64T Long Mode**

Modus panjang (long mode) adalah sebuah modus prosesor 64-bit IA32e (x86-64/AMD64/x64/EM64T) yang berjalan di atas sistem operasi 64-bit, sehingga ia dapat mengeluarkan seluruh kemampuannya, seperti halnya mengakses memori lebih besar daripada 4 GB (hingga batasan yang dimiliki oleh prosesor dan sistem operasi), dan menjalankan aplikasi 64-bit. Hanya beberapa sistem operasi yang dapat menjalankan prosesor IA32e dalam modus ini, yakni Windows XP Professional x64 Edition, Windows Server 2003, GNU/Linux (versi kernel2.6 ke atas), Solaris 10 dan beberapa varian UNIX lainnya.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Generation** | **First introduced** | **Prominent consumer CPU brands** | **Linear/physical address space** | **Notable (new) features** |
| 1 | 1978 | Intel 8086, Intel 8088 and clones | **16-bit** / 20-bit (segmented) | First x86 microprocessors |
| 1982 | Intel 80186, Intel 80188 and clones, NEC V20/V30 | Hardware for fast address calculations, fast mul/div, etc. |
| 2 | Intel 80286 and clones | **16-bit** (30-bit virtual) / 24-bit (segmented) | MMU, for protected mode and a larger address space. |
| 3 (IA-32) | 1985 | Intel 80386 and clones, AMD Am386 | **32-bit** (46-bit virtual) / 32-bit | 32-bit instruction set, MMU with paging. |
| 4 (FPU) | 1989 | Intel486 and clones, AMD Am486/Am5x86 | RISC-like pipelining, integrated x87 FPU (80-bit), on-chip cache. |
| 4/5 | 1997 | IDT/Centaur-C6, Cyrix III-Samuel, VIA C3-Samuel2 / VIA C3-Ezra (2001), VIA C7 (2005) | In-order, integrated FPU, some models with on-chip L2 cache, MMX, SSE. |
| 5 | 1993 | Pentium, Pentium MMX, Cyrix 5x86, Rise mP6 | Superscalar, 64-bit databus, faster FPU, MMX (2�- 32-bit). |
| 5/6 | 1996 | AMD K5, Nx586 (1994) | μ-op translation. |
| 6 | 1995 | Pentium Pro, Cyrix 6x86, Cyrix MII, Cyrix III-Joshua (2000) | *As above* / **36**-bit physical (PAE) | μ-op translation, conditional move instructions, Out-of-order, register renaming, speculative execution, PAE (Pentium Pro), in-package L2 cache (Pentium Pro). |
| 1997 | AMD K6/-2/3, Pentium II/III | L3-cache support, 3DNow!, SSE (2�- 64-bit). |
| 2003 | Pentium M, Intel Core (2006) | optimized for low power. |
| 7 | 1999 | Athlon, Athlon XP | Superscalar FPU, wide design (up to three x86 instr./clock). |
| 2000 | Pentium 4 | deeply pipelined, high frequency, SSE2, hyper-threading. |
| 7/8 | 2000 | Transmeta Crusoe, Efficeon | VLIW design with x86 emulator, on-die memory controller. |
| 2004 | Pentium 4 Prescott | **64-bit** / 40-bit physical in first AMD implementation | Very deeply pipelined, very high frequency, SSE3, 64-bit capability (integer CPU) is available only in LGA 775 sockets. |
| 2006 | Intel Core 2 | 64-bit (integer CPU), low power, multi-core, lower clock frequency, SSE4 (Penryn). |
| 2008 | VIA Nano | Out-of-order, superscalar, 64-bit (integer CPU), hardware-based encryption, very low power, adaptive power management. |
| 8 (x86-64) | 2003 | Athlon 64, Opteron | x86-64 instruction set (CPU main integer core), on-die memory controller, hypertransport. |
| 8/9 | 2007 | AMD Phenom | *As above* / 48-bit physical for AMD Phenom | Monolithic quad-core, SSE4a, HyperTransport 3 or QuickPath, native memory controller, on-die L3 cache, modular. |
| 2008 | Intel Core i3/i5/i7, AMD Phenom II |
| Intel Atom | In-order but highly pipelined, very-low-power, on some models: 64-bit (integer CPU), on-die GPU. |
| 2011 | AMD Bobcat, Llano | Out-of-order, 64-bit (integer CPU), on-die GPU, low power (Bobcat). |
| 9 (GPU) | 2011 | Intel Sandy Bridge/Ivy Bridge, AMD Bulldozer and Trinity |  | SSE5/AVX (4�- 64-bit), highly modular design, integrated on-die GPU. |
| 2013 | Intel Haswell |  | AVX2 and FMA3 instructions. |
| — (MIC pilot) | 2012 | Intel Xeon Phi (Larrabee) |  | Many Integrated Cores (62), In-order P54C with x86-64, Very wide vector unit, LRBni instructions (8�- 64-bit) |

**Set intruksi 8051 dibagi menjadi 3 kelompok besar, yaitu :**

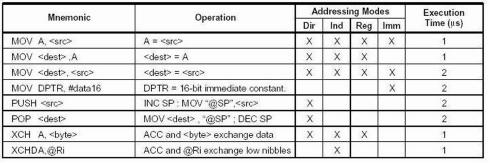
1. Intruksi-intruksi transfer data

2. Intruksi-intruksi pemrosesan data

3. Intruksi-intruksi lompatan

**1. Intruksi-intruksi transfer data**

1.1. Daftar Intruksi Transfer data

[](https://raw.githubusercontent.com/L200150043/Organisasi-dan-Arsitektur-Komputer/master/img/tugas01-table.jpg)

1.2. Instruksi Transfer Data

a) RAM Internal

Perintah perpindahan data (MOV, XCH, POP, PUSH) pada RAM internal membutuhkan 1 sampai 2 cycle. Format instruksi :

MOV (tujuan), (asal)

Memungkinkan data untuk berpindah diantara 2 lokasi RAM internal atau SFR tanpa harus melalui akumulator terlebih dahulu.

b) RAM EKternal

Perintah MOV 16-bit digunakan untuk inisialisasi DPTR atau untuk akses data 16-bit pada memori ekternal.perpindahan data antera memori internal dan ekternal menggunakan indirect addressing dengan menggunakan alamat 1-byte (@R1) atau 2-byte (@DPTR).

c) Look Up Tables

Ada dua perintah untuk membaca look-up tables pada ROM. MOVC (move constant) menggunakan program counter sebagai base register dan akumulator sebagai offsetnya.

MOVC A, @A+DPTR

Perintah tersebut dapat mengakses 256 entri. Nomor entri dimasukkan ke akumulator dan awal tabelnya pada DPTR.

MOVC A, @A+PC

**2. Instruksi-intruksi Pemrosesan Data**

Instruksi-intruksi pemrosesan data dibagi manjadi 2 :

1. Instruksi-intruksi Logika

2. Intruksi-intruksi Aritmatika

Set intruksi Aritmatika terdiri dari :

- Instruksi-instruksi penjumlahan dan pengurangan

- Instruksi-instruksi perkalian dan pembagian

- Instruksi-instruksi Increment dan Decrement

- Instruksi pengubahan ke decimal (Decimal Adjust)

| **Instruction** | **Meaning** | **Notes** | **Opcode** |
| --- | --- | --- | --- |
| AAA | ASCII adjust AL after addition | used with unpacked binary coded decimal |  |
| AAD | ASCII adjust AX before division | 8086/8088 datasheet documents only base 10 version of the AAD instruction (opcode 0xD5 0x0A), but any other base will work. Later Intel's documentation has the generic form too. NEC V20 and V30 (and possibly other NEC V-series CPUs) always use base 10, and ignore the argument, causing a number of incompatibilities |  |
| AAM | ASCII adjust AX after multiplication | Only base 10 version is documented, see notes for AAD |  |
| AAS | ASCII adjust AL after subtraction |  |
| ADC | Add with carry | destination := destination + source + carry\_flag |  |
| ADD | Add | (1) r/m += r/imm; (2) r += m/imm; |  |
| AND | Logical AND | (1) r/m &= r/imm; (2) r &= m/imm; |  |
| CALL | Call procedure | push eip + 2 ; jmp operand |  |
| CBW | Convert byte to word |  |  |
| CLC | Clear carry flag | CF = 0; |  |
| CLD | Clear direction flag | DF = 0; |  |
| CLI | Clear interrupt flag | IF = 0; |  |
| CMC | Complement carry flag |  |  |
| CMP | Compare operands |  |  |
| CMPSB | Compare bytes in memory |  |  |
| CMPSW | Compare words |  |  |
| CWD | Convert word to doubleword |  |  |
| DAA | Decimal adjust AL after addition | (used with packed binary coded decimal) |  |
| DAS | Decimal adjust AL after subtraction |  |  |
| DEC | Decrement by 1 |  |  |
| DIV | Unsigned divide | DX:AX = DX:AX / r/m; resulting DX = remainder |  |
| ESC | Used with floating-point unit |  |  |
| HLT | Enter halt state |  | 0xF4 |
| IDIV | Signed divide | DX:AX = DX:AX / r/m; resulting DX = remainder |  |
| IMUL | Signed multiply | (1) DX:AX = AX \* r/m; (2) AX = AL \* r/m |  |
| IN | Input from port | (1) AL = port[imm]; (2) AL = port[DX]; (3) AX = port[DX]; |  |
| INC | Increment by 1 |  |  |
| INT | Call to interrupt |  |  |
| INTO | Call to interrupt if overflow |  |  |
| IRET | Return from interrupt |  |  |
| Jcc | Jump if condition | (*JA, JAE, JB, JBE, JC, JCXZ, JE, JG, JGE, JL, JLE, JNA, JNAE, JNB, JNBE, JNC, JNE, JNG, JNGE, JNL, JNLE, JNO, JNP, JNS, JNZ, JO, JP, JPE, JPO, JS, JZ*) |  |
| JMP | Jump |  |  |
| LAHF | Load flags into AH register |  |  |
| LDS | Load pointer using DS |  |  |
| LEA | Load Effective Address |  |  |
| LES | Load ES with pointer |  |  |
| LOCK | Assert BUS LOCK# signal | (for multiprocessing) |  |
| LODSB | Load string byte | if (DF==0) AL = \*SI++; else AL = \*SI--; |  |
| LODSW | Load string word | if (DF==0) AX = \*SI++; else AX = \*SI--; |  |
| LOOP/LOOPx | Loop control | (*LOOPE, LOOPNE, LOOPNZ, LOOPZ*) if (x && --CX) goto lbl; |  |
| MOV | Move | copies data from one location to another, (1) r/m = r; (2) r = r/m; |  |
| MOVSB | Move byte from string to string | if (DF==0) \*(byte\*)DI++ = \*(byte\*)SI++; else \*(byte\*)DI-- = \*(byte\*)SI--; |  |
| MOVSW | Move word from string to string | if (DF==0) \*(word\*)DI++ = \*(word\*)SI++; else \*(word\*)DI-- = \*(word\*)SI--; |  |
| MUL | Unsigned multiply | (1) DX:AX = AX \* r/m; (2) AX = AL \* r/m; |  |
| NEG | Two's complement negation | r/m \*= -1; |  |
| NOP | No operation | opcode equivalent to XCHG EAX, EAX |  |
| NOT | Negate the operand, logical NOT | r/m ^= -1; |  |
| OR | Logical OR | = r/imm; (2) r |= m/imm; |  |
| OUT | Output to port | (1) port[imm] = AL; (2) port[DX] = AL; (3) port[DX] = AX; |  |
| POP | Pop data from stack | \*SP++ = r/m; POP CS (opcode 0x0F) works only on 8086/8088. Later CPUs use 0x0F as a prefix for newer instructions. | 0x0F |
| POPF | Pop data from flags register | \*SP++ = flags; |  |
| PUSH | Push data onto stack | \*--SP = r/m; |  |
| PUSHF | Push flags onto stack | \*--SP = flags; |  |
| RCL | Rotate left (with carry) |  |  |
| RCR | Rotate right (with carry) |  |  |
| REPxx | Repeat MOVS/STOS/CMPS/LODS/SCAS | (*REP, REPE, REPNE, REPNZ, REPZ*) |  |
| RET | Return from procedure | either RETN or RETF depending on memory model of assembler etc. |  |
| RETN | Return from near procedure |  |  |
| RETF | Return from far procedure |  |  |
| ROL | Rotate left |  |  |
| ROR | Rotate right |  |  |
| SAHF | Store AH into flags |  |  |
| SAL | Shift Arithmetically left (signed shift left) | (1) r/m <<= 1; (2) r/m <<= CL; |  |
| SAR | Shift Arithmetically right (signed shift right) | (1) (signed)r/m >>= 1; (2) (signed)r/m >>= CL; |  |
| SBB | Subtraction with borrow | alternative 1-byte encoding of SBB AL, AL is available via undocumented SALC instruction |  |
| SCASB | Compare byte string |  |  |
| SCASW | Compare word string |  |  |
| SHL | Shift left (unsigned shift left) |  |  |
| SHR | Shift right (unsigned shift right) |  |  |
| STC | Set carry flag | CF = 1; |  |
| STD | Set direction flag | DF = 1; |  |
| STI | Set interrupt flag | IF = 1; |  |
| STOSB | Store byte in string | if (DF==0) \*ES:DI++ = AL; else \*ES:DI-- = AL; |  |
| STOSW | Store word in string | if (DF==0) \*ES:DI++ = AX; else \*ES:DI-- = AX; |  |
| SUB | Subtraction | (1) r/m -= r/imm; (2) r -= m/imm; |  |
| TEST | Logical compare (AND) | (1) r/m & r/imm; (2) r & m/imm; |  |
| WAIT | Wait until not busy | Waits until BUSY# pin is inactive (used with floating-point unit) |  |
| XCHG | Exchange data | r :=: r/m; |  |
| XLAT | Table look-up translation | behaves like MOV AL, [BX+AL] |  |
| XOR | Exclusive OR | (1) r/m ^= r/imm; (2) r ^= m/imm; | </p>  </td>  </tr>  </tbody> |