**INSTRUCTION SET UNTUK PROCESSOR INTEL**

**DAN ARM**

**Arsitektur Microprocessor 8088**

            Di awali dari definisi Mikroprosesor itu sendiri, ialah sebuah chip (IC) yang bekerja dengan program. Fungsinya adalah sebagai pengontrol atau pengolah utama dalam suatu rangkaian elektronik. [Mikroprosesor](http://de-kill.blogspot.com/2009/05/sekilas-sejarah-mikroprosesor.html) biasa disebut juga dengan CPU (Central Processing Unit). Sekilas tentang sejarahnya yaitu pada tahun 1978  Intel mengeluarkan mikroprosesor 16-bit yaitu 8086, setahun kemudian mengeluarkan 8088 dengan kecepatan eksekusi dan memori lebih besar dari 8085, serta mulai digunakannya cache memori (sistem antrian yang mengatur pemberian instruksi sebelum menjalankannya.

            Mikroprosesor Intel 8088 termasuk keluarga mikroprosesor 8 bit dan 16 bit. Mikroprosesor 8088 mempunyai 8 bit jalur data dan 20 bit jalur alamat. Jalur data memiliki pin yang sama dengan jalur alamat, artinya pada saat tertentu digunakan sebagai jalur data dan pada saat yang lain digunakan sebagai jalur alamat.

            Setiap satu pin memiliki dua fungsi, yaitu sebagai jalur data dan jalur alamat maka digunakanlah sistem time multiplexing, yaitu penggunaan jalur yang sama untuk fungsi dan waktu yang berbeda, sehingga tidak bisa data dan alamat dikirim pada saat yang bersamaan.

            Mikroprosesor 8088 mampu mengalamati memori sampai 1 MB. Memori ini digunakan untuk menyimpan kode biner dari instruksi yang akan dijalankan oleh μP, selain itu memori juga digunakan untuk menyimpan data secara sementara.

**Set Instruksi Pada Microprocessor 8088**

**1. Transfer Data**

Set instruksi 8088 mencakup 14 transfer data yang menggerakkan byte atau kata data antara memori dan register selain juga antara akumulator dan port I/O.

a. General Purpose (Umum) : MOV, PUSH, POP, XCHG,SXLAT

b. Input/Output                               : In, Out

c. Address Transfer (Pemindahan Alamat)   : LEA, KDS, KES

d. Flag Transfer (Pemindahan Flag)                 : LAHF, SAHF, PUSHF, POPF

**2. Arithmetic Instructions (Instruksi Perhitungan)**

8088 mampu melakukan penambahan, pengurangan, pengurangan dan pembagian data selain juga bytes dan kata. Sistem ini menambah dan mengurangi dengan menggunakan bytes atau kata yang bertanda atau tak bertanda dan data BCD atau ASCII.

a. Addition (Penjumlahan) : ADD, ADC, AAA, DAA, INC

b. Substraction (Pengurangan) : SUB, SBB, AAS, DAS, DES, NEG, CMP

c. Multiplication (Perkalian)     : MUL, IMUL, AAM

d. Division (Pembagian)              : DIV, IDIV, AAD

e. Sign Extension                        : CBW, CWD

**3. Manipulasi Bit**

Instruksi ini mencakup operasi logika, shift dan rotasi.

a. Logical (Logika)          : AND, OR, XOR, NOT, TEST

b. Shift (Geser)                 : SAL/SHL, SAR, SHR

c. Rotate (Gulung) : ROL, ROR, RCL, RCR

**4. Instruksi String**

Instruksi string digunakan untuk memanipulasi string data dalam memori. Setiap string tersusun baik dari bytes maupun kata dan hingga mencapai 64 K bytes panjang.nya

a. Direction           : CLD, STD

b. Repeat Prefixes           : REP, REPE/REPZ, REPNE/REPNZ,

c. Move                        : MOVSB, MOVSW

d. Compare                   : CMPS, CMPSB, CMPSW

e. Scan                           : CSAS, SCASB, SCASW

d. Load and Store            : LODS, LODSB, LODSW, STOS, STOSB, STOSW

**5. Transfer Program**

Instruksi transfer program mencakup*jump,  call*, dan *return*, instruksi yang sudah kita kenal dalam 8085.

a. Unconditional transfer (Pemindahan)         : CAL, RET, JMP

b. Conditional transfer (Pemindahan Bersyarat)         : JA/JNBE, JAE/JNB, JB/JNAE/JC, JBE/JNA, JCXZ, JE/JZ, JG/JNLE, JGF/JNL, JL/JNGE, JLE/JNG, JNC, JNE/JNZ, JNO, JNP/JPO, JNS, JO, JP/JPE, JS

c. Loop/Jump             : LOOP, LOOPE/LOOPZ, LOOPNE/LOOPNZ

**6. Kontrol Prosesor**

Instruksi kontrol prosesor memungkinkan dan tidak memungkinkan interupsi, memodifikasi bit flag dan mensinkronisasi kejadian eksternal.

a. Flag Operations : CLC, STC, CMC, CLD, STD, CLI, STI,

b. External Synchronization

**7. Interrupt**  : INT, INTO, IRET

**Berikut ini penjelasan tentang instruksi yang digunakan pada bahasa rakitan 8086 dan 8088 atau sejenisnya :**

1. **AAA** (ASCII adjust for edition) Pengaturan ASCII bagi penambahan
2. **AAD**(ASCII adjust for division) Pengaturan ASCII bagi pembagian
3. **AAM** (ASCII adjust for multipy) Pengaturan ASCII bagi perkalian
4. **AAS**(ASCII adjust for substraction) Pengaturan ASCII bagi pengurangan

5.  **ADC** (add with carry) T ambahkan dengan carry

6.  **ADD** (addition) Penambahan

7.  **AND**(logic AND) Logik AND

8.  **CALL**(CALL subroutine) Subrutin panggil

9.  **CBW** (Convert byte to word) Konversikan byte ke kata

10. **CLC**(Clear Carry) Kosongkan corry flag

11. **CLD**(Clear directon flag) Kosongkan flag arah

12. **CLI** (Clear interrupt enable) Kosongkan flag penggerak interupsi

13. **DAS** (decimal adjust for substraction) Pengaturan decimal bagi pengurangan

14. **DEC** (decrement) Penurunan operand tujuan dengan 1

15. **DIV**(divide) Pembagian tak bertanda

16. **ESC** (escape) Escape digunakan sehubungan dengan suatu co-procesor external

17. **HLT**(halt) HLT menghentikan prosesor sampai saluran reset diaktifkan

18. **IDIV** (Integer division) Pembagian kilat

19. **IMUL** (Integer multiply) Perkalian kilat

20. **IN** (Input) IN (masukan) mentransfer data dari port yang disperifikasikan ke dalam register AC atau AX

21. **NC** (Increment) INC menaikkan operand tujuan dalam 1

22. **INT** (Interupt) INT (Interupsi) mengawali suatu prosedur interupsi dengan jenis yang dispesifikasikan oleh instruksi yang bersangkutan

23. **INTO** (Interupt on overflow) Instrupsi bila ada overflow INTO digunakan untuk membangkitkan suatu interupsi perangkat lunak yang bergantumg pada status flag OF

24. **IRET** (Interupt return) IRET digunakan untuk kembali dari suatu interupsi dan mendapatkan kembali IP,CS dan flag-flag dari stack

25. **JA**(Jump on above) Digunakan jika operand pertama lebih besar daripada operand kedua

26. **JNBE**(Jump) Digunakan jika pada saat pembandingan operand 1, tidak lebih kecil atausama dengan operand 2

27. **JAE** (Jump on above or equal) Digunakan pada saat pembandingan operand 1 lebih besar atau sama dengan operand 2

28. **JNB** (Jump not below) Digunakan pada saat pembandingan operand 1 tidak lebih kecil dari operand 2

29. **JB** (Jump on below) Digunakan pada saat operand 1 lebih kecil dari operand 2

30. **JNAE** (Jump on not above or equal) Digunakan pada saat operand 1 tidak lebih besar atau sama dengan operand 2

31. **JBE** (Jump on below or equal) Digunakan pada saat operand 1 lebih kecil atau sama dengan operand 2

32. **JNA** (Jump on not above) Digunakan pada saat operand 1 tidak lebih besar dari operand 2

33. **JC** (Jump on carry) Digunakan pada saat akan diprogram CF=1

34. **JCXZ** (Jump if CX=0) Digunakan bila isi register CX=0

35. **JE**(Jump on equal) Digunakan pada saat pembandingan kedua operand sama

36. **JZ** (Jump on zero) Digunakan pada saat akan diproses ZF=1

37. **JG** (Jump on greather than) Digunakan pada saat pembandingan operand 1 menunjukkan lebih besar dari operand 2

38. **JNLE** (Jump on greather or equal) Digunakan pada saat operand 1tidak lebih kecil atau sama dengan operand 2

39. **JGE** (Jump on greather or equal) Digunakan pada saat operand 1 lebih besar atau sama dengan operand 2

40. **JNL** (Jump on less) Digunakan pada saat operand 1 tidak lebih kecil dari operand 2

41. **JL** (Jump on less) Digunakan pada saat operand 1 lebih kecil dari operand 2

42. **JNGE** (Jump on not greather or equal) Digunakan pada saat operand 1 tidak lebih besar atau sama dengan operand 2

43. **JLE** (Jump on less or equal) Digunakan pada saat operand 1 lebih kecil atau sama dengan operand 2

44. **JNG** (Jump on not greather than) Digunakan pada saat operand 1 tidak lebih besar daripada operand 2

45. **JMP** (Unconditional jump) Lompatan tidak bersyarat

46. **JNC**(Jump on not carry) Digunakan pada saat diproses CF=O

47. **JNE** (Jump on not equal) Digunakan pada saat operand 1 tidak sama dari operand 2

48. **JNO** (Jump on not zero) Digunakan pada saat diproses ZF=0

49. **JNO** (Jump on not overflow) Digunakan bila tidak ada overflow (OF=O)

50. **JNS**(Jump on not sign) Digunakan pada saat diproses SF=O

51. **ZNP** (Jump on not pority) Digunakan bila tidak ada pointer

52. **JPO** (Jump on poity odd) Digunakan bila pointer ganjil

53. **O** (Jump on overflow) Digunakan pada saat ada overflow

54. **JP** (Jump on Pority equal) Digunakan bila pointer genap

55. **JS** (Jump on sign) Digunakan pada saat diproses ZF=1

56. **LAHF** (Load AH from flag) LAHF me-load bit 7,6,4,2 & 0 pada register AH masing-masing dengan isi flag SF.ZF,AF & CF

57. **LDS**(Load pointer using DS) Muatkan peminjak dengan menggunakan DS

58. **LEA** (Load effective address) LEA mentransfer opsan sumber 16 bit dalam memori ke tujuan 16 bit

59. **LES**(Load pointer using ES) Muatkan pointer dengan menggunakan ES

60. **LOCK** (Lock bas) Digunakan dalam penerapan pemakaian sumber bersama, untuk memastikan bahwa memori tidak diakses secara serentak oleh lebih dari satu proses

61. **LODS**(Load string) Muatkan string byte atau kata

62. **LOOP**Loop jika EX bukan 0

63. **LOOPE** dan LOOPZ -Loop bila sama

64. **-Loop**bila 0

65. **LOOPNE/LOOPNZ**-Loop bila tidak sama (ZF=0)

66. **-Loop**bila tidak nol (ZF=O)

67. **MOVE** Pindahkan byte atau kata memindahkan data 8 bit atau 16 bit

68. **MOUS** (Move string) Pindahkan data string 8 bit atau 16 bit

69. **MUL**(Multifly) Digunakan untuk mengalihkan isi bertandapada akumulator dengan suatu operand sumber yang dispesifikasikan

70. **NEG**(Negate) Mengurangkan suatu operand tujuan dari 0 & menyimpan hasil komplemen keduanya dalam tujuan

71. **NOP**(Logic NOT) NOP tidak membe rikan dampak tertentu (tidak ada operasi)

72. **NOT**(Logic OR) Operasi logic OR

73. **OR** (Logic OR) Operasi logic OR

74. **OUT** (Output) Digunakan untuk mentransfer data dari AL atau AX ke point (pangkalan)

75. **POP** (POP from stack) Keluaran data dari stack dari AL atau AX ke point (pangkalan)

76. **POPF**(POP flag) Keluarkan ke flag dan stack

77. **POSH** (Push to stack) Dorong sumber ke stack

78. **PUSHF**(Push flag) Dorong flag ke stack

79. **RCL** (Rotate lift with carry) Putar ke kirin dengan carry 1

80. **RCR** (Rotate right with carry) Putar ke kanan dengan carry 1

81. **REP**Ulangi

82. **REE**Ulangi jika sama

83. **REPZ** Ulangi jika nol

84. **REPNE** Ulangi jika tidak sama

85. **REPNZ** Ulangi jika tidak nol

86. **RET** (Return from subroutine) Digunakan untuk kembali ke subrutin

87. **ROL**(Rotate left) Rotasi (putar) ke kiri 1

88. **ROR** (Rotate right) Rotasi (putar) ke kanan 1

89. **SAHF** (store AH in flags register) Simpan AH dalam reegister flag

90. **SAL/SHL**(Shift left arithmetic/logical) Geser aritmatik/logika ke kiri 1

91. **SAR/SHR**(Shift right arithmetic/logical) Geser aritmatik/logika ke kanan 1

92. **SBB** (Substract with borrow) Kurangkan dengan borrow

93. **SCAS** (Scan string) Digunakan untuk mengurangkan string tujuan yang dialamatkan a/n regiser DI,dari AX atau AL

94. **STC** (Set carry flag) Menset carry flag (CF)

95. **STD** (Set direction flag) Menset flag arah (DF)

96. **STI** (Set interrupt enable flag) Menset flag penggerak interupsi (IF)

97. **STOS**(Substract) Memindahkan operand sumber yang terkandung dalam AX atau AL ke suatu tujuan yang dialamatkan oleh DI

98. **SUB** (Substract) Mengurangkan sumber dari operand tujuan dan hasilnya ditepatkan dalam tujuan

99. **TEST**(Logical comporison) Menguji operand-operand atau logic perbandingan

100. **WAIT** Menunggu sampai saluran test aktif

101. **XCHG** (Exchange) Pertukaran dan sumber

102. **XLAT** (Translate) Digunakan untuk menerjemahkan se-byte dalam register AL ke dalam suatu byte yang diambil dari tabel terjemahan

103. **XOR** (Logic exclusive OR) Operasi exclusive OR

**SEKILAS INTEL 8088**

Intel 8088 adalah prosesor mikro buatan Intel berbasis pada 8086, dengan 16-bit register dan menggunakan 8-bit external data bus. intel 8088 merupakan prosesor yang digunakan pada IBM PC.

8088 ditargetkan pada sistem yang ekonomis, diikuti oleh pengunaan desain 8-bit. Jalur bus yang lebar dalam circuit boards masih sangatlah mahal ketika ini di luncurkan. Queue yang ungul dari 8088 adalah 4 bytes, sebagai penggunaan dalam 8086 6 bytes. 8088 termasuk keturunan dari 80188, 80288, 80186, 80286, 80386, 80486, dan 80388, microcontroller seperti yang masih digunakan sekarang. Clone yang populer dengan menggunakan 8088 adalah Model D, dimana tombol pilihan dapat berjalan pada clock 4.77 MHZ atau 7.16 MHZ.

**Spesifikasi 8088**

Mikroprosesor Intel 8088 hampir serupa dengan prosesor Intel 8086, kecuali pada data  eksternal bus. Lebar data eksternal bus 8088 dikurangi menjadi 8 bit, dan instruksi ukuran queue dan prefetching algoritmanya diubah. Intel 8088 menggunakan dua urutan bus siklus untuk menulis atau membaca 16 data bit sebagai ganti satu siklus untuk 8086. Ini menjadikan prosesor  bergerak  lebih lambat, tetapi ada nilai plus pada perangkat keras yang menjadikan CPU 8088 kompatibel dengan peripheral 8080/8085.

Pin SSO pada 8088 menggantikan BHE/S7 pada 8086, dan pin IO/M pada 8088, bukan M/IO seperti pada 8086. 8088 membutuhkan catu daya +5,0 V dengan toleransi + 10%. Mikroprosesor 8088 akan kompatibel TTL (Transistor-Transistor Logic) jika kekebalan terhadap noise disesuaikan menjadi 350 mV dari nilai 400 mV yang biasa. Mikroprosesor 8088 dapat menjalankan satu 74XX, lima 74LSXX, satu 74SXX, sepuluh 74ALSXX, dan sepuluh 74HCXX beban satuan. Jika mikroprosesor 8088 direset, mikroprosesor ini mulai mengeksekusi perangkat lunak pada lokasi meori FFFF0H (FFF:0000) dengan pin interrupt request di-disable. Karena bus-bus 8088 dimultipleks dan kebanyakan memori dan peralatan I/O tidak, system harus didemultipleks sebelum pengantarmukaan dengan memori atau I/O. Demultipleks dilakukan oleh latch delapan bit yang pulsa clocknya didapat dari sinyal ALE. Operasi minimum 8088 sama dengan mikroprosesor Intel 8085A, sementara mode maksimum adalah baru dan khusus dirancang untuk operasi koprosesor aritmatika 8087.

**REGISTER**

Sebuah register adalah sebuah tempat penampungan sementara untuk data-data yng akan diolah oleh prosesor, dan dibentuk oleh 16 titik elektronis di dalam chip mikroprosesor itu sendiri. Dengan adanya tempat-tempat penampungan data sementara ini, proses pengolahan akan bisa dilakukan

secara jauh lebih cepat dibandingkan apabila data-data tersebut harus diambil langsung dari lokasi-lokasi memori. Register-register tersebut sebagai register internal dan terdiri dari empat belas register dan keseluruhannya dapat dibagi dalam beberapa jenis, yaitu : Register Segment, Register Data, Register pointer, Register index, Register index, dan General Purpose Register.

Semua general register mikroprosesor  8088 dapat digunakan untuk perhitungan dan operasi logika.

Pada 8088, register data diwujudkan oleh AX, BX, CX dan BX (sebagai general purpose register), sehubungan dengan fungsinya yang selain menangani tugas-tugas khusus, juga bisa dimanfaatkan untuk membantu proses-proses pengolahan data didalam internal mikroprosesor.

Mikroprosesor 8088 mempunyai kemampuan untuk bekerja dalam mode 'langkah tunggal' (single-step), yaitu semua instruksi dilaksanakan dengan cara satu demi satu. Mode ini dimungkinkan

dengan jalan membuat TF (Trap Flag) masuk ke logika '1' atau 'set'. Bagi seorang programmer, mode ini akan sangat berguna dalam pekerjaan 'debugging'.

Organisasi Ruang Memori Dan Register

Unsur terkecil memori adalah 'sel memori' (memory-cell), yaitu suatu elemen penyimpanan data yang berkapasitas sebesar 1 bit. Dengan menggabungkan sejumlah sel memori, akan bisa membentuk suatu ruang penyimpanan data dengan berbagai ukuran, misalnya 1 byte, 1 word, 1 Kilobyte, 1 Megabyte, 1 Gigabyte, 1 Terabyte, dsb.

Organisasi memori dapat dibandingkan dengan sistem 'locker' (susunan laci yang mempunyai kode nomor setiap lacinya sehingga memudahkan orang mengenal lacinya masing-masing sebelum mengambil atau memasukkan barang titipannya).

Susunan chip prosesor Intel 8088.

• Catu Daya/VCC (pin 40) dan GND (pin 1 dan 20)

• Bus Data (AD0 - AD7)

• Bus Alamat (AD0 - AD7 dan A8 - A19)

• Bus Kendali (NMI, INTR, CLK, Reset).

Dalam mikroprosesor 8088 secara fisik, bus alamat terdiri dari 20 bit (A0-A19), sementara register-register internal terbentuk dari 16 bit data. Oleh sebab itu, untuk menyesuaikan perbedaan jumlah bit antara bus alamat 8088 dengan register internal, sistem pengalamatan memori dilaksanakan

dengan format segment:offset. Format yang membutuhkan 32 bit ini dibentuk dengan jalan menggabungkan data dari 2 buah register sekaligus. Register pertama adalah satu satu dari 4 register segment, sedangkan register lainnya diambil dari salah sebuah register pointer atau register indeks.

Kenyataannya, segment-segment yang didefinisikan pada ruang memori itu boleh dibuat saling berdampingan, terpisah atau tumpang tindih sekalipun. Prosesor memiliki bus alamat sebanyak 20 bit, yang berarti ia mampu mengalamati hingga 1.048.575 lokasi memori. Secara heksadesimal, jumlah ini dinyatakan sebagai angka 00000 sampai  dengan FFFFF.  Ini adalah alamat-alamat fisik (physical addresses) dari mikroprosesor. Untuk 8088 dan 8086 yang bus alamatnya terdiri dari 20 bit, otomatis penulisan alamat fisiknya terdiri dari 5 digit heksadesimal. Sistem segmentasi pada IBM PC dilaksanakan agak unik. 1 segment adalah bagian dari ruang memori yang besarnya 65536 byte atau 64 Kb. Namun, segment-segment itu tidaklah diletakkan secara berdampingan sambung menyambung satu sama lain, akan tetapi saling tumpang tindih sehingga jarak antara titik awal suatu segment hanya terpaut 16 byte terhadap segment lainnya.

Peta Memori (Memory Map)

Kapasitas memori untuk IBM PC/XT yang berbasis prosesor Intel 8088/8086 adalah 1.048.576 byte atau lebih mudah disebut 1 (satu) Megabyte. Nilai sebesar 1 MB inilah yang menjadi dasar sistem pemetaan memori dalam keluarga IBM PC Kompatibel, sehingga dalam produk-produk yang lebih mutakhir pun, peta memori tersebut tetap dipertahankan. Hal ini berhubungan dengan konsistensi yang harus dijaga pada Disk Operating System, yang dalam keadaan bagaimanapun, harus tetap bisa dijalankan mulai dari produk yang paling awal seperti PC/XT, sampai kepada yang terbaru seperti AT 486 kompatibel.

**ARSITEKTUR**

Arsitektur dari 8088 tetap sama degan 8086 yakni: 16-bit registers, 16-bit internal data bus dan 20-bit address bus, yang bisa menjadikan prosesor mencapai memori 1 MB. 8088 memiliki pembagian memori yang sama dengan 8086: prosesor bisa mencapai 64 KB dari memori secara langsung, dan untuk mencapai lebih dari 64 KB, salah satu dari bagian khusus register harus di update.

Program, data dan stack memori menduduki ruang memori yang sama. Total kapasitas memory yang bisa dicapai adalah 1MB KB. Sebagaimana kebanyakan instruksi prosesor yang menggunakan 16-bit pointers, prosesor dapat mengolah secara efektif jika hanya memorinya 64 KB. Untuk mengakses memori diluar 64 KB,  CPU menggunakan bagian  register khusus untuk menspesifikasi di mana kode, stack dan 64 KB segmen data diposisikan di dalam memori 1 MB.

16-bit pointers dan data disimpan sebagai:

address: low-order byte

address+1: high-order byte

32-bit addresses disimpan di "segment:offset" dengan format:

address: low-order byte of segment

address+1: high-order byte of segment

address+2: low-order byte of offset

address+3: high-order byte of offset

Physical memory address ditunjukkan oleh pasangan segment:offset dihitung dengan:

address = (<segment> \* 16) + <offset>

Program Memori- program dapat ditempatkan di manapun di dalam memori. perintah jump and call dapat digunakan untuk menyingkat lompatan di dalam segmen kode 64 KB, seperti halnya untuk lompatan jauh di manapun di dalam memori 1 MB. Seluruh perintah lompatan yang bersyarat dapat digunakan untuk melompat sekitar + 127 - - 127 bytes dari instruksi yang ada.

Memori data - prosesor dapat mengakses data di tiap orang lebih dari 4 segmen yang tersedia, yang membatasi ukuran dari memori yang dapat diakses sampai 256 KB ( jika seluruh empat segmen menunjuk pada 64 KB blok berbeda). Mengakses data dari Data, Code, segmen Extra atau Stack biasanya dapat dilaksanakan dengan awalan perintah DS:, CS:, SS: atau ES: ( beberapa register dan instruksi dengan tak hadir boleh gunakan segmen SS atau ES sebagai ganti segmen DS).

**Set Instruksi**

Set instruksi 8088  terdiri dari perintah-perintah berikut:

 \* Instruksi perpindahan data.

 \* Aritmatika - penjumlahan, pengurangan, penaikan, penurunan, mengkonversi byte/word dan pembandingan.

 \* Logika - DAN, OR, eksklusif OR, shift/rotate dan test.

 \* Manipulasi string - load, store, move, compare dan scan untuk byte/word.

 \* Kontrol transfer - conditional, unconditional, panggilan subroutine dan kembali dari subroutine.

 \* Perintah Input/Output.

 \* Lain-lain - setting/clearing flag bits, stack operations, software interrupts, dll.

### CISC vs RISC

Pada saat ini ada dua konsep populer yang berhubungan dengan desain CPU dan set instruksi. Keduanya adalah:

1. Complex Instruction Set Computing (CISC)
2. Reduce Instruction Set Computing (RISC).

Semua sistem yang lama (komputer mainframe, komputer mini atau komputer mikro) relatif mempunyai sistem CISC. Walaupun sistem sekarang terdiri atas kedua jenis tersebut. Sistem RISCsaat ini lebih populer karena tingkat kerjanya, dibandingkan dengan sistem CISC. Namun karena biayanya tinggi, sistem RISC hanya digunakan ketika diperlukan kecepatan khusus, keandalan dan sebagainya.

#### CISC (Complex Instruction-Set Computer)

Complex Instruction Set Computer (CISC) adalah sebuah arsitektur dari set instruksi dimana setiap instruksi akan menjalankan beberapa operasi tingkat rendah, seperti pengambilan dari memory, operasi aritmetika, dan penyimpanan ke dalam memory, semuanya sekaligus hanya di dalam sebuah instruksi. Tujuan utama dari arsitektur CISC adalah melaksanakan suatu perintah cukup dengan beberapa baris bahasa mesin sedikit mungkin. Hal ini bisa tercapai dengan cara membuat perangkat keras prosesor mampu memahami dan menjalankan beberapa rangkaian operasi. Untuk tujuan contoh kita kali ini, sebuah prosesor CISC sudah dilengkapi dengan sebuah instruksi khusus, yang kita beri nama MULT. Saat dijalankan, instruksi akan membaca dua nilai dan menyimpannya ke 2 register yag berbeda, melakukan perkalian operan di unit eksekusi dan kemudian mengambalikan lagi hasilnya ke register yang benar.

Karakteristik CISC yg "sarat informasi" ini memberikan keuntungan di mana ukuran program-program yang dihasilkan akan menjadi relatif lebih kecil, dan penggunaan memory akan semakin berkurang. Karena CISC inilah biaya pembuatan komputer pada saat itu (tahun 1960) menjadi jauh lebih hemat. Memang setelah itu banyak desain yang memberikan hasil yang lebih baik dengan biaya yang lebih rendah, dan juga mengakibatkan pemrograman level tinggi menjadi lebih sederhana, tetapi pada kenyataannya tidaklah selalu demikian.   
Contoh-contoh prosesor CISC adalah : System/360, VAX, PDP-11, varian Motorola 68000 , dan CPU AMD dan Intel x86. 

CISC mempunyai karakteristrik :

1. Instruksi berukuran tunggal
2. Ukuran yang umum adalah 4 byte.
3. Jumlah mode pengalamatan data yang sedikit, biasanya kurang dari lima buah
4. Tidak terdapat pengalamatan tak langsung
5. Tidak terdapat operasi yang menggabungkan operasi load/store dengan operasi aritmetika (misalnya, penambahan dari memori, penambahan ke memori)

#### RISC (Reduce Instruction Set Computer)

RISC Reduced Instruction Set Computingatau "Komputasi set instruksi yang disederhanakan. Merupakan sebuah arsitektur komputer atau arsitektur komputasi modern dengan instruksi-instruksi dan jenis eksekusi yang paling sederhana. Biasanya digunakan pada komputer berkinerja tinggi seperti komputer vektor.  
Bahasa pemprograman memungkinkan programmer dapat mengekspresikan algoritma lebih singkat, lebih memperhatikan rincian, dan mendukung penggunaan pemprograman terstruktur, tetapi ternyata muncul masalah lain yaitu semantic gap, yaitu perbedaan antara operasi-operasi yang disediakan oleh HLL dengan yang disediakan oleh arsitektur komputer, ini ditandai dengan ketidakefisienan eksekusi, program mesin yang berukuran besar,dan kompleksitas kompiler.   
Set-set instruksi yang kompleks tersebut dimaksudkan untuk :

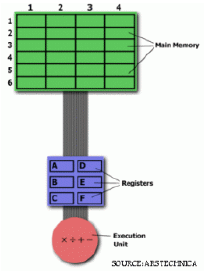
1. Memudahkan pekerjaan kompiler
2. Meningkatkan efisiensie ksekusi, karena operasi yang kompleks dapat diimplementasikan didalam mikrokode.
3. Memberikan dukungan bagi HLL yang lebih kompleks dan canggih

RISC mempunyai karakteristik :

1. One cycle execution time : satu putaran eksekusi.
2. Prosessor RISC mempunyai CPI (clock per instruction) atau waktu per instruksi untuk setiap putaran. Hal ini dimaksud untuk mengoptimalkan setiap instruksi pada CPU.
3. Pipelining adalah sebuah teknik yang memungkinkan dapat melakukan eksekusi secara simultan. Sehingga proses instruksi lebih efiisien.
4. Large number of registers: Jumlah register yang sangat banyak.
5. RISC didesain dimaksudkan untuk dapat menampung jumlah register yang sangat banyak untuk mengantisipasi agar tidak terjadi interaksi yang berlebih dengan memory.
6. Rangkaian instruksi built-in pada processor yang terdiri dari perintah-perintah yang lebih ringkas dibandingkan dengan CISC.
7. RISC memiliki keunggulan dalam hal kecepatannya sehingga banyak digunakan untuk aplikasi-aplikasi yang memerlukan kalkulasi secara intensif.

#### Perbandingan antara RISC dengan CISC

Cara sederhana untuk melihat kekurangan dan kelebihan dari CISC dan RISC adalah dengan membandingkannya secara langsung. Pada tahap perbandingan ini dicoba dengan menghitung perkalian dua bilangan dalam memori. Memori tersebut terbagi menjadi beberapa lokasi yang diberi nomor 1(baris): 1 (kolom) hingga 6:4. Unit eksekusi bertanggung-jawab untuk semua operasi komputasi. Namun, unit eksekusi hanya beroperasi untuk data yang sudah disimpan kedalam salah satu dari 6 register (A, B, C, D, Eatau F). Misalnya, kita akan melakukan perkalian (product) dua angka, satu di simpan di lokasi 2:3 sedangkan lainnya di lokasi 5:2, kemudian hasil perkalian tersebut dikembalikan lagi kelokasi 2:3

[](http://3.bp.blogspot.com/--xp5SlpISPA/VCfl-kGXWcI/AAAAAAAAAD4/39mCo4Y73D8/s1600/CISC%2Bdan%2BRISC.png)

1. **Menggunakan Pendekatan RISC**

Prosesor RISC hanya menggunakan instruksi-instruksi sederhana yang bisa dieksekusi dalam satu siklus. Dengan demikian, instruksi ‘MULT’ sebagaimana dijelaskan sebelumnya dibagi menjadi tiga instruksi yang berbeda, yaitu“LOAD”, yang digunakan untuk memindahkan data dari memori kedalam register, “PROD”, yang digunakan untuk melakukan operasi produk (perkalian) dua operan yang berada di dalam register (bukan yang ada di memori) dan “STORE”, yang digunakan untuk memindahkan data dari register kembali kememori. Berikut ini adalah urutan instruksi yang harus dieksekusi agar yang terjadi sama dengan instruksi “MULT” pada prosesor RISC (dalam 4 baris bahasa mesin):

**LOAD A, 2:3   
LOAD B, 5:2   
PROD A, B   
STORE 2:3, A**

1. **Menggunakan Pendekatan CISC**

Tujuan utama dari arsitektur CISC adalah melaksanakan suatu perintah cukup dengan beberapa baris bahasa mesin sedikit mungkin. Hal ini bisa tercapai dengan cara membuat perangkat keras prosesor mampu memahami dan menjalankan beberapa rangkaian operasi.  
Sebuah prosesor CISC sudah dilengkapi dengan sebuah instruksi khusus, yang diberi nama MULT. Saat dijalankan, instruksi akan membaca dua nilai dan menyimpannya ke 2 register yang berbeda, melakukan perkalian operan di unit eksekusi dan kemudian mengambalikan lagi hasilnya ke register yang benar. Jadi instruksi-nya cukup satu saja.

**MULT 2:3, 5:2**

**Perbedaan CISC dan RISC:**

**CISC:**

* Penekanan pada perangkat keras (hardware).
* Termasuk instruksi kompleks multi-clock.
* Memori-ke-memori: “LOAD” dan “STORE” saling bekerjasama..
* Ukuran kode kecil, kecepatan rendah.
* Transistor digunakan untuk menyimpan instruksi-instruksi kompleks.

**RISC**

* Penekanan pada perangkat lunak (software).
* Single-clock, hanya sejumlah kecil instruksi.
* Register ke register: “LOAD” dan “STORE” adalah instruksi - instruksi terpisah.
* Ukuran kode besar, kecepatan (relatif) tinggi.
* Transistor banyak dipakai untuk register memori.

### x86 vs ARM

#### 1.  x86

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.

1. **Mode Real (Real Mode)**

Intel 8086 dan 8088 dilengkapi dengan 14 16-bit register. Empat diantaranya (AX, BX, CX, DX) dirancang sebagai fungsi umum (general purpouse) (meskipun masing-masing juga memiliki fungsi khusus tambahan; misalnya hanya register CX yang dapat digunakan sebagai penghitung (counter) dalam instruksi loop). Setiap register dapat diakses sebagai dua byte (8-bit) terpisah (jadi byte-atas BX’s dapat diakses sebagai BH dan byte-bawah-nya sebagai BL). Selain itu, terdapat juga empat register segmen (CS, DS, SS dan ES). Register ini digunakan untuk membangun alamat memori. Ada juga dua register penunjuk (pointer) (SP yang menunjuk pada titik awal stack, dan BP yang dapat menunjuk pada titik manapun dalam stack atau memori). Ada dua register indeks (SI dan DI) yang dapat digunakan sebagai penunjuk dalam array. Dan terakhir, ada sebuah register penanda (register flag) yang terdapat didalamnya penanda-penanda seperti carry, overflow, zero dan lain-lain, dan juga sebuah penunjuk instruksi (instruction pointer – IP) yang menunjuk ke alamat instruksi yang sedang dieksekusi.

Dalam mode real, memori diakses secara tersegmentasi. Hal ini dilakukan dengan menggeser (shifting) alamat segmen 4 bit ke kiri dan menambah sebuah ofset untuk menghasilkan alamat akhir sepanjang 20-bit. Contohnya, jika DS berisi nilai A000h dan SI berisi nilai 5677h, DS:SI akan mengacu pada titik alamat real DS × 16 + SI = A5677h. Jadi jumlah total alamat memori yang dapat diakses dalam mode real adalah 220 byte, atau 1 MiB, jumlah yang sangat mengesankan di tahun 1978. Seluruh alamat memori terbagi dalam segmen dan ofset; dan setiap tipe akses (kode, data, atau stack) memiliki register segmen tertentu (untuk data register yang digunakan DS, untuk kode digunakan register CS, dan untuk stack digunakan SS). Untuk mengakses data, register segmen dapat secara langsung dipilih (dengan melakukan ubah-paksa (override) prefik segmen) dari empat register segmen yang tersedia.

Dengan aturan ini, dua pasang segmen/ofset yang berbeda bisa mengacu ke lokasi memori absolut yang sama. Jadi bila DS berisi A111h dan SI 4567h, DS:SI akan menunjuk ke alamat A56777h seperti di atas. Lebih lanjut, CS dan SS berperan vital bagi program agar berfungsi secara benar, sehingga hanya DS dan ES yang dapat dipakai untuk mengacu ke segmen data diluar program (atau lebih tepatnya, diluar segmen program yang sedang dieksekusi) atau stack. Skema ini, yang semula bertujuan mempertahankan kompatibilitas dengan Intel 8085, sering dikeluhkan oleh para programer (walaupun beberapa programer tidak terlalu mempedulikannya, dan popularitas x86 sebelum tahun-tahun mode proteksi diperkenalkan membuktikan bahwa hal ini bukan cacat yang sangat serius).

Selain itu, 8086 juga memiliki 64 KB alamat I/O 8-bit (atau 32 K-word dari 16-bit), dan satu 64 KB (satu segmen) stack di memori yang didukung oleh hardware (memakai register-register SS, SP, dan BP). Hanya word (2 byte) yang bisa di-push ke stack. Stack tumbuh ke bawah (ke arah alamat yang lebih rendah secara numerik), dengan ujung bawah diacu oleh SS:SP. Ada 256 interrupt yang dapat diaktifkan oleh hardware maupun software. Interrupt tersebut bisa bertingkat, memanfaatkan stack untuk menyimpan alamat balik.

CPU x86 32-bit yang modern masih mendukung real mode, bahkan memulai operasi pada real mode setelah reset. Kode real mode yang dijalankan pada prosesor-prosesor tersebut bisa memanfaatkan register 32-bit dan register segmen tambahan (FS dan GS) yang mulai tersedia sejak 80386.

**Mode terproteksi 16-bit (16-bit protected mode)**

Prosesor 80286 dapat menjalankan Operating System yang menggunakan Mode Real 16-bit prosesor 8086 tanpa perubahan pada OS, namun prosesor ini juga mempunyai mode lain, yaitu mode terproteksi. Mode terproteksi memungkinkan program untuk mengakses RAM yang berukuran 16MB, dan memiliki memori virtual hingga 1GB. Hal ini dimungkinkan karena mode terproteksi menggunakan register segmen untuk menyimpan index ke sebuah tabel segmen. Pada 80286, terdapat dua buah tabel segmen, yaitu tabel GDT dan tabel LDT, masing-masing dapat menyimpan hingga 8192 perinci segmen, tiap segment memberi akses untuk memori sebesar hingga 64KB. Tabel segmen ini menyimpan alamat dasar yang berukuran 24-bit, yang akan digunakan untuk menghitung alamat absolut dari memori yang akan digunakan. Selain itu, segmen-segmen ini dapat diberikan salah satu dari empat jenis level akses.

Walaupun dibuatnya prosesor ini merupakan pengembangan yang baik, prosesor ini sangat jarang digunakan karena mode terproteksi tidak dapat menjalankan program-program yang berjalan mode real yang sudah ada sebagai proses, karena program-program mode real sering mengakses perangkat keras secara langsung dan beberapa ada yang melakukan aritmatika segmen, sehingga tidak dapat dijalankan pada mode terproteksi.

#### 2. ARM

Prosesor berasitektur ARM merupakan prosesor yang paling banyak digunakan dalam perangkat mobile saat ini. Desainnya yang sederhana, membuat prosesor ARM cocok untuk aplikasi berdaya rendah.

Bebeda dari Intel dan AMD yang memproduksi dan menjual prosesor rancangan mereka sendiri. ARM hanya menjual lisensi hak paten desain prosesor mereka kepada berbagai perusahaan manufaktur semikonduktor.

Lisensi arsitektur proseor ARM saat ini dimiliki oleh Alcatel, Atmel, Broadcom, Cirrus Logic, Digital Equipment Corporation, Freescale, Intel melalui DEC, LG, Marvell Technology Group, NEC, NVIDIA, NXP Semiconductors, OKI, Quallcomm, Samsung, Sharp, ST Microelectronics, Symbios Logic, Texas Instruments, VLSI Technology, Yamah dan ZiiLABS. Dalam situs resmi perusahaan, ARM mengklaim telah menjual 30 miliar unit prosesor dan lebih dari 16 juta prosesor terjual setiap harinya.

Pada awalnya, Advanced RISC Machines (ARM) yang didirikan pada tahun 1990 ini merupakan sebuah perusahaan patungan antara Acorn Computers, Apple Computer (sekarang Apple Inc), dan VLSI Technology. Pada tahun 1998, Advanced RISC Machines berganti nama menjadi ARM Holdings. Perusahaan ini fokus pada penelitan dan pengembangan desain arsitektur prosesor.

Arsitektur ARM menjadi dasar bagi sebagian besar central processing unit (CPU) di berbagai perangkat mobile. ARM mempunyai desain arsitektur Cortex seri M, R, A, hingga seri A57. Mereka juga memiliki desain arsitektur untuk graphics processing unit (GPU) bernama Mali.

Menurut data dari ARM Holding, arsitektur ARM menguasai 95 persen arsitektur prosesor smartphone, 35 persen televisi digital, serta 10 persen komputer mobile. Hampir semua tablet dan smartphone unggulan, baik yang berbasis Android, iOS, BlackBerry, hingga Windows Phone, memakai prosesor arsitektur ARM. Beberapa konsol game portabel, seperti Game Boy Advance, Nintendo DS, dan PlayStation Portable, memakai prosesor ARM.

ARM juga mengekspansi bisnisnya ke pasar komputer server. Pada Mei 2012, Dell mengumumkan Copper, server yang memakai prosesor Marvell dengan desain ARM.

Pada Oktober 2012, ARM juga mengumumkan prosesor 64-bit dari keluarga Cortex-A57 yang saat ini digunakan oleh iPhone 5s, iPad Air dan iPad Mini generasi kedua. Samsung juga telah membeli lisensi desain ARM 64-bit untuk smartphonenya yang akan dirilis di 2014.

|  |  |
| --- | --- |
| **Prosessor X86** | **Prosessor ARM** |
| * Arsitektur ARM merupakan arsitektur [prosesor](http://id.wikipedia.org/wiki/Prosesor) 32-bit RISC yang dikembangkan oleh *ARM Limited.* Dikenal sebagai *Advanced RISC Machine* dimana sebelumnya dikenal sebagai *Acorn RISC Machine*. Pada awalnya merupakan *prosesor desktop*yang sekarang didominasi oleh keluarga x86.      * Desainnya  yang sederhana membuat prosesor ARM cocok untuk aplikasi berdaya rendah.Prosesor ARM digunakan di berbagai bidang seperti elektronik umum, termasuk PDA, mobile phone, media player, music player, game console genggam, kalkulator dan periperal komputer seperti hard disk drive dan router.      * prosesor yang menggunakan perintah sederhana misal diberi  perintah menghitung 24 ARM akan mengartikan perintah 24  sebagai 2\*2\*2\*2 * prosesor RISC (ARM)cenderung memiliki transistor jauh lebih sedikit daripada prosesor CISC dengan  transistor  yang lebih sedikit  itu berarti  daya yang  digunakan juga lebih kecil, lebih sedikit panas, dan ruang yang sangat kecil, sehingga sangat cocok untuk perangkat-perangkat kecil dengan kemepuan terbatas.      * operasi  floating point dan operasi pembagian tidak termasuk dalam mode pengalamatan prosessor RISC(ARM) operasi  floating point dann operasi pembagian  biasanya diturunkan untuk coprocessors. * Proyek Acorn RISC Machine resmi dimulai pada Oktober 1983. VLSI Technology, Inc dipilih sebagai mitra dalam memproduksi chip silikon dimana sebelumnya telah memproduksi ROM dan custom chip sebelumnya. Proses desain dipimpin oleh Wilson dan Furber, dengan tujuan utama latensi rendah (low-latency) pada penanganan input/output (interupsi) seperti pada prosesor MOS Technology 6502. Arsitektur 6502 memberikan pengembang mesin yang cepat dalam pengaksesan memory tanpa harus menggunakan perangkat direct access memory yang mahal. VLSI memproduksi chip ARM pertama kali pada 26 April 1985 yang berhasil bekerja dan dikenal sebagai ARM1. Dan disusul dengan ARM2 yang diproduksi pada tahun berikutnya dan terus berkembang. * Pada akhir 1980-an, Apple Computer dan VLSI Technology memulai kerja sama dengan Acorn untuk prosesor ARM berikutnya. Hasil kerja sama Apple-ARM menghasilkan ARM6 pada awal tahun 1992. Apple menggunakan ARM6 (ARM 610) sebagai prosesor pada PDA Apple Newton dan pada tahun 1994 Acorn menggunakan ARM6 pada komputer PC RISC. Pada frekuensi 233 MHz, prosesor ini hanya mengonsumsi daya sebesar 1 Watt dan versi berikutnya lebih kecil dari itu.      * Arsitektur x86 adalah rancangan [Set Instruksi Komputer Kompleks (Complex Instruction Set Computer)](http://id.wikipedia.org/wiki/CISC) dengan panjang instruksi yang bervariasi. prosesor-prosesor awal dari keluarga arsitektur ini memiliki nomor model yang diakhiri dengan urutan angka “86″: prosesor [8086](http://id.wikipedia.org/wiki/8086), [80186](http://id.wikipedia.org/wiki/80186),[80286](http://id.wikipedia.org/wiki/80286), [386](http://id.wikipedia.org/wiki/Intel_80386), dan [486](http://id.wikipedia.org/wiki/Intel_80486). Karena nomor tidak bisa dijadikan merek dagang, Intel akhirnya menggunakan kata [Pentium](http://id.wikipedia.org/wiki/Pentium)untuk merek dagang processor generasi kelima mereka. | * Arsitektur x86 saat ini mendominasi komputer desktop, komputer portabel, dan pasar server sederhana. Bisa menjalankan program windows karena memang Windows di program untuk beroprasi di lingkungan x86.      * prosesor yang menggunakan perintah rumit misal diberi  perintah menghitung 24×86 akan  akan mengartikan 24 sebagai 2^4…. * prosesor CISC(x86) memiliki transistor yang lebih banyak otomatis daya  yang dibutuhkan lebih besar, dan tidak menutup kemungkinan prosesor yang memiliki banyak transistor akan lebih cepat panas,serta membutuhkan memeori yang mencukupi(besar) sehingga tidak cocok untuk perangkat-perangkat kecil      * operasi floating point dan operasi pembagian termasuk dalam mode pengalamatan prosessor CISC(x86)      * Arsitektur x86 pertama kali hadir melalui [8086](http://id.wikipedia.org/wiki/Intel_8086) [CPU](http://id.wikipedia.org/wiki/Central_processing_unit) pada tahun [1978](http://id.wikipedia.org/wiki/1978); Intel 8086 adalah pengembangan dari mikroprosesor [Intel 8080](http://id.wikipedia.org/wiki/Intel_8080) (yang dibangun mengikuti arsitektur dari [4004](http://id.wikipedia.org/wiki/Intel_4004) dan [8008](http://id.wikipedia.org/wiki/Intel_8008)), dan program bahasa rakitan dari 8080 dapat diterjemahkan secara mekanik ke program yang setara ke bahasa rakitan untuk 8086. Arsitektur ini diadaptasi (dengan versi yang lebih sederhana dari versi[8088](http://id.wikipedia.org/wiki/Intel_8088)) tiga tahun kemudian sebagai standar dari CPU pada[IBM PC](http://id.wikipedia.org/wiki/IBM_PC). Kehadiran platform PC secara luas membuat arsitektur x86 menjadi arsitektur CPU yang paling sukses selama ini. (Rancangan CPU lainnya yang sangat sukses, yang dibagun berdasarkan 8080 dan kompatible pada set-instruksi hingga pada tingkatan bahasa-mesin biner adalah arsitektur[Zilog Z80](http://id.wikipedia.org/wiki/Zilog_Z80).)      * Arsitektur ini telah dua kali diperluas untuk mengakomodasi [ukuran word](http://id.wikipedia.org/w/index.php?title=Word_%28istilah_komputer%29&action=edit&redlink=1)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](http://id.wikipedia.org/w/index.php?title=IA-32&action=edit&redlink=1) (singkatan dari Intel Architecture, 32-bit). Kemudian pada tahun 2003,[AMD](http://id.wikipedia.org/wiki/AMD) memperkenalkan [Athlon 64](http://id.wikipedia.org/wiki/Athlon_64), yang menerapkan secara lebih jauh pengembangan dari arsitektur ini menuju ke arsitektur 64-bit, dikenal dengan beberapa istilah [x86-64](http://id.wikipedia.org/wiki/AMD64), [AMD64](http://id.wikipedia.org/wiki/AMD64) (AMD), [EM64T](http://id.wikipedia.org/w/index.php?title=EM64T&action=edit&redlink=1)atau [IA-32e](http://id.wikipedia.org/w/index.php?title=IA-32e&action=edit&redlink=1) (Intel), dan x64 (Microsoft). |