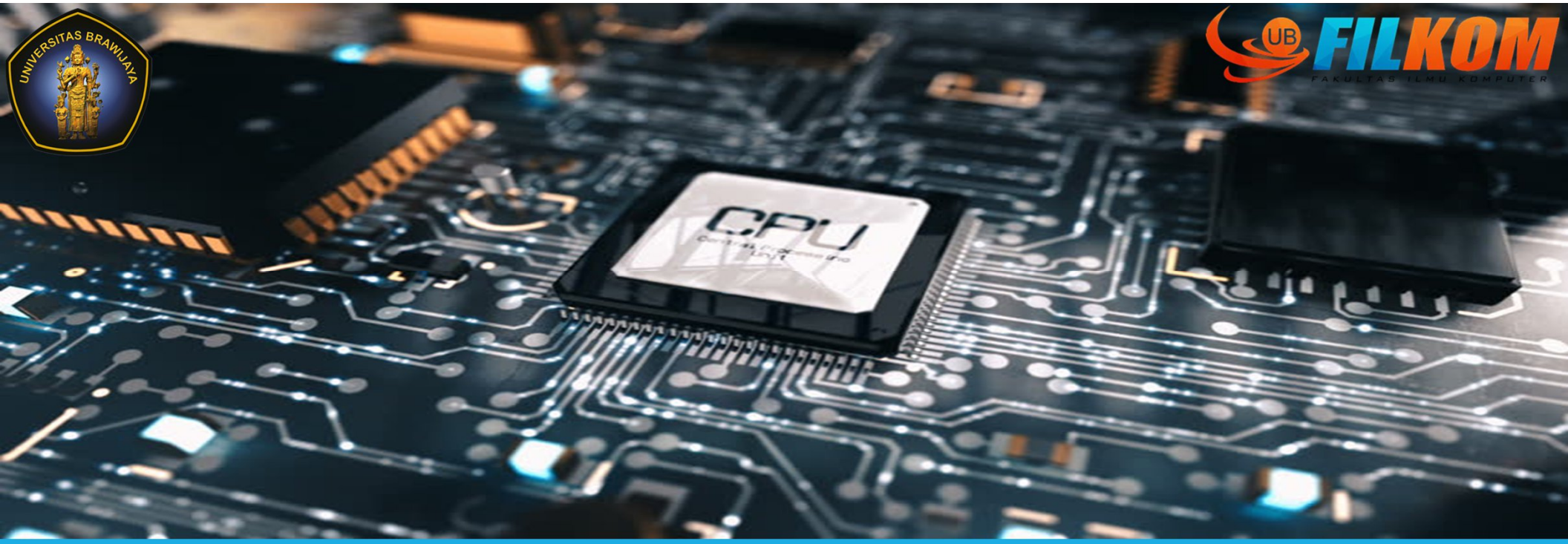


Arsitektur dan Organisasi Komputer

COM 60011

Topik #10 – Instruction Set Architecture: Karakteristik dan Fungsi

Pendahuluan

- **Tujuan**
  - Menjelaskan gambaran umum karakteristik penting dari instruksi mesin
  - Menjelaskan jenis operand yang digunakan dalam set instruksi mesin khusus.
  - Menjelaskan gambaran umum tipe data x86 dan ARM.
  - Menjelaskan jenis operand yang didukung oleh set instruksi mesin.
  - Menjelaskan gambaran umum jenis operasi x86 dan ARM.
  - Memahami perbedaan antara big-endian dan little-endian
- **Materi**
  - Karakteristik instruksi mesin
  - Jenis operand
  - Tipe data Intel x86 dan ARM
  - Jenis operasi
  - Jenis operasi Intel x86 dan ARM
  - Big-endian dan little-endian

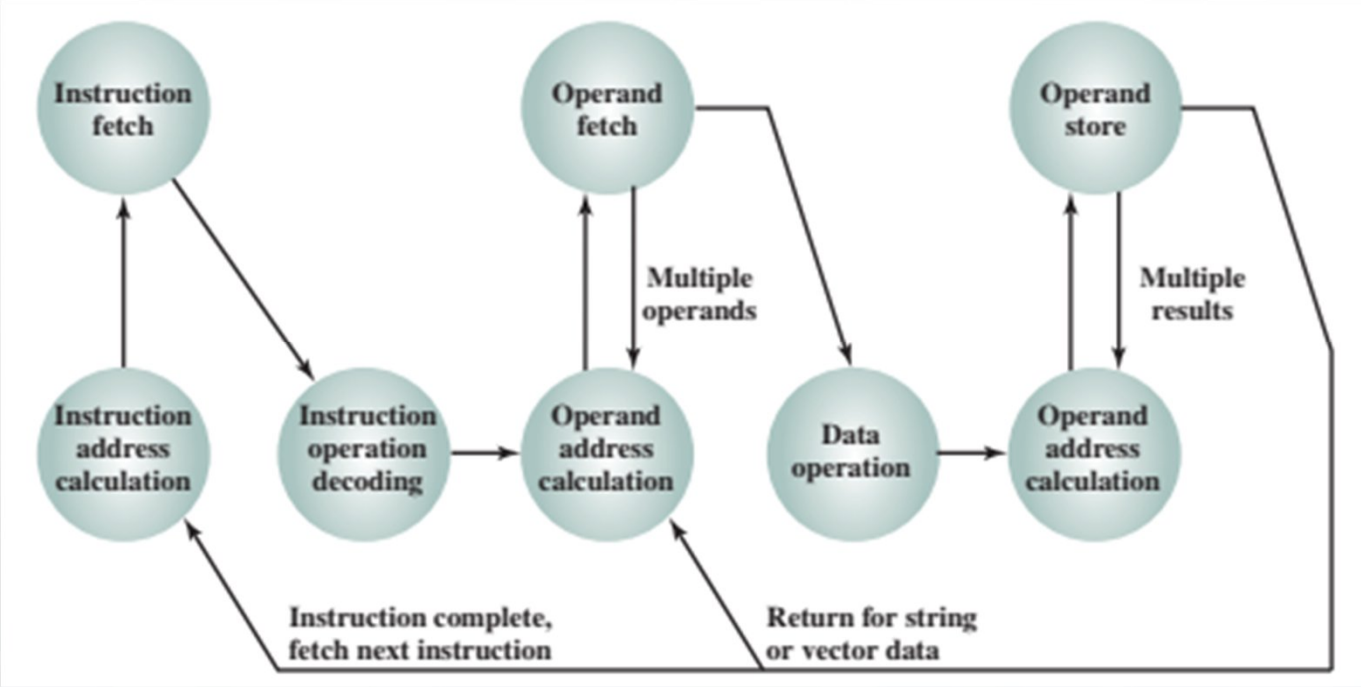
Karakteristik Instruksi Mesin

- Instruksi mesin atau instruksi komputer → operasi prosesor yang ditentukan oleh instruksi yang dijalankan
- Set instruksi → kumpulan instruksi pada prosesor yang dapat dieksekusi
- Setiap instruksi harus berisi informasi yang diperlukan oleh prosesor untuk eksekusi

Elemen Instruksi Mesin

- **Operation Code (Opcode) / Kode Operasi**
  - Menentukan operasi yang akan dilakukan, seperti ADD, I/O
  - Ditentukan oleh kode biner
    - Kerjakan ini
- **Source Operand Reference / Alamat Asal Operand**
  - Operasi melibatkan satu atau lebih operand asal, yaitu operand yang merupakan input untuk operasi.
    - Terhadap isi alamat ini
- **Result Operand Reference / Alamat Hasil Operand**
  - Hasil atau output operasi
    - Letakkan hasilnya di alamat ini
- **Next Instruction Reference / Alamat Instruksi Berikutnya**
  - Memberi informasi ke prosesor posisi atau alamat untuk mengambil instruksi berikutnya setelah eksekusi selesai

Diagram State Siklus Instruksi

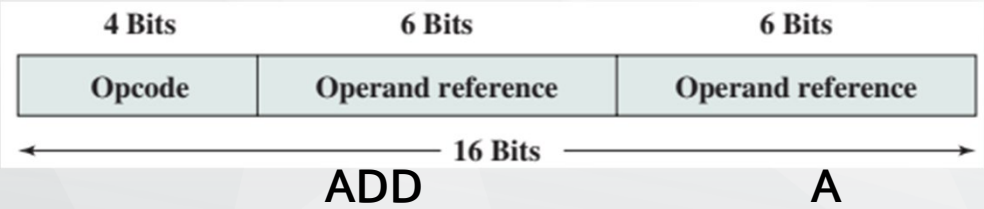


Area Operand Hasil dan Operand Asal

- **Memori virtual atau utama**
  - Seperti pada referensi instruksi berikutnya
- **Register prosesor**
  - Prosesor berisi satu atau lebih register yang dapat direferensikan oleh instruksi mesin
  - Jika ada lebih dari satu register, setiap register diberi nama atau nomor unik dan instruksi harus berisi nomor register yang diinginkan
- **Immediate**
  - Nilai operand terdapat pada field instruksi yang sedang dieksekusi
- **Perangkat I/O**
  - Instruksi harus menentukan modul I / O dan perangkat untuk operasi tersebut. Jika memory-mapped I/O digunakan, hal ini hanyalah alamat memori utama atau virtual lainnya

Representasi Instruksi

- **Di dalam komputer, setiap instruksi direpresentasikan oleh urutan bit**
- **Instruksi dibagi menjadi beberapa field, sesuai dengan elemen penyusun instruksi**
- **Misal instruksi dengan 2 alamat operand**
  - ADD A, B → A dan B adalah suatu alamat register



- **Representasi opcode dengan singkatan → mnemonics**
  - Memiliki representasi biner tetap
  - Programmer menentukan lokasi setiap operand simbolik

Contoh mnemonics:

ADD	Add
SUB	Subtract
MUL	Multiply
DIV	Divide
LOAD	Load data dari memori
STOR	Simpan data ke memori

Jenis Instruksi

- **Data processing**
  - Instruksi aritmatika (ADD, SUB, dsb) dan logika (AND, OR, NOT, dsb)
- **Data storage**
  - Pergerakan data masuk atau keluar dari register dan atau lokasi memori
  - STOR, LOAD, MOVE, dsb
- **Data movement**
  - Instruksi I/O
- **Control**
  - Flow control
  - Instruksi percabangan dan test
  - JUMP, HALT, dsb

Format Instruksi 3 Alamat

- **Bentuk umum:**
  - [OPCODE] [Result Operand], [Source Operand 1], [Source Operand 2]
- **Contoh: ADD Y, A, B**
  - Bentuk algoritmik: Y ← A + B
  - Tambahkan isi register A dengan isi register B, kemudian simpan hasilnya di register Y
- **Mengoperasikan banyak register sekaligus, oleh karena itu bentuk ini tidak umum digunakan di komputer**
- **Program lebih pendek**

Contoh Format 3 Alamat

- **A, B, C, D, E, T, Y merupakan register**
- **Program: Y = (A – B) / (C + D x E)**

Instruksi		Komentar
SUB	Y, A, B	Y ← A – B
MPY	T, D, E	T ← D x E
ADD	T, C, T	T ← C + T
DIV	Y, Y, T	Y ← Y / T

- **Memerlukan 4 instruksi**

Format Instruksi 2 Alamat

- **Bentuk umum:**
  - [OPCODE] [Result Operand], [Source Operand]
  - Satu result operand merangkap operand, satu alamat operand
- **Contoh: ADD Y, B**
  - Bentuk algoritmik: Y ← Y + B
  - Tambahkan isi register Y dengan isi register B, kemudian simpan hasilnya di register Y
- **Instruksi ini masih digunakan di komputer sekarang**
- **Mengoperasikan lebih sedikit register, tapi panjang program tidak bertambah terlalu banyak**

Contoh Format 2 Alamat

- A, B, C, D, E, T, Y merupakan register
- Program: Y = (A – B) / (C + D x E)

Instruksi		Komentar
MOVE	Y, A	Y ← A
SUB	Y, B	Y ← Y – B
MOVE	T, D	T ← D
MPY	T, E	T ← T x E
ADD	T, C	T ← T + C
DIV	Y, T	Y ← Y / T

- Memerlukan 6 operasi

Format Instruksi 1 Alamat

- **Bentuk umum:**
  - [OPCODE] [Source Operand]
  - Satu source operand, hasil disimpan di accumulator
- **Contoh: ADD B**
  - Bentuk algoritmik: AC ← AC + B
  - Tambahkan isi accumulator dengan isi register B, kemudian simpan hasilnya di accumulator
- **Instruksi ini digunakan pada komputer jaman dahulu**
- **Hanya mengoperasikan satu register, tapi program bertambah panjang**

Contoh Format 1 Alamat

- **A, B, C, D, E, Y merupakan register**
- **Program: Y = (A – B) / (C + D x E)**

Instruksi		Komentar
LOAD	D	AC ← D
MPY	E	AC ← AC x E
ADD	C	AC ← AC + C
STOR	Y	Y ← AC
LOAD	A	AC ← A
SUB	B	AC ← AC - B
DIV	Y	AC ← AC / Y
STOR	Y	Y ← AC

Memerlukan 8 operasi

Format Instruksi 0 Alamat

- **Bentuk umum:**
  - [OPCODE] [Source Operand]d]
  - Semua alamat operand implisit, disimpan dalam bentuk stack. Operasi yang biasanya membutuhkan 2 operand, akan mengambil isi stack paling atas dan di bawahnya
- **Contoh: ADD**
  - Bentuk algoritmik: S[top] ← S[top-1] + S[top]
  - Tambahkan isi stack no.2 dari atas dengan isi stack paling atas, kemudian simpan hasilnya di stack paling atas
- **Ada instruksi khusus stack: PUSH dan POP yang dapat diberi alamat**

Contoh Format 0 Alamat

- A, B, C, D, E, Y merupakan register
- Program: Y = (A – B) / (C + D x E)

Instruksi		Komentar
PUSH	A	S[top] ← A
PUSH	B	S[top] ← B
SUB		S[top] ← A – B
PUSH	C	S[top] ← C
PUSH	D	S[top] ← D
PUSH	E	S[top] ← E
MPY		S[top] ← D x E
ADD		S[top] ← C + S[top]
DIV		S[top] ← (A – B) / S[top]
POP	Y	Out ← S[top]

Memerlukan 10 operasi

Jenis Operand

- **Instruksi mesin beroperasi pada data.**
- **Kategori umum yang penting dari data:**
  - Alamat → dijelaskan di Bab berikutnya
  - Bilangan
  - Karakter
  - Data logikal
- **Beberapa kalkulasi harus dilakukan pada operand reference dalam instruksi untuk menentukan alamat memori utama atau virtual**
  - Alamat dapat dianggap sebagai bilangan unsigned integer

Operand Bilangan

- **Semua bahasa mesin termasuk jenis data numerik**
- **Bilangan yang disimpan di komputer adalah terbatas**
  - Batas besaran bilangan direpresentasikan dalam mesin
  - Batas presisi pada bilangan floating-point
- **3 jenis data numerik yang umum di komputer**
  - Binary integer atau binary fixed point
  - Binary floating-point
  - Desimal
- **Packed desimal**
  - Setiap digit desimal direpresentasikan oleh kode 4 bit dengan dua digit yang disimpan per byte
  - Untuk merangkai kode bilangan 4 bit, biasanya dalam kelipatan 8 bit

Operand Karakter

- **Bentuk data yang umum → string karakter dan teks**
- **Data tekstual dalam bentuk karakter tidak dapat dengan mudah disimpan atau ditransmisikan oleh pemrosesan data dan sistem komunikasi, karena dirancang untuk data biner**
- **Kode karakter yang umum digunakan → International Reference Alphabet (IRA)**
  - di Amerika Serikat disebut sebagai American Standard Code for Information Interchange (ASCII)
- **Kode lain yang digunakan untuk encode karakter adalah Extended Binary Coded Decimal Interchange Code (EBCDIC)**
  - Digunakan pada komputer mainframe IBM

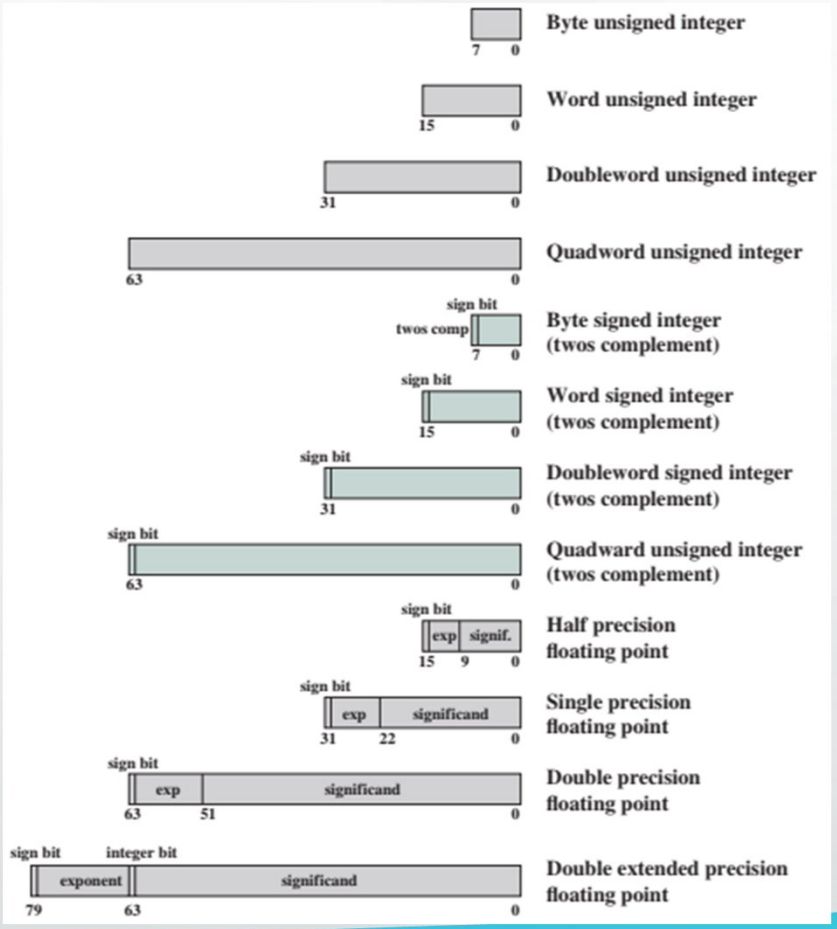
Data Logikal

- **Unit n-bit yang terdiri dari data n 1-bit, setiap item memiliki nilai 0 dan 1**
- **Dua keuntungan dari tampilan bit-oriented:**
  - Memori efisien menyimpan array Boolean atau item data biner → nilai 1 (true) dan 0 (false)
  - Untuk manipulasi bit item data
    - Jika operasi floating-point diimplementasikan dalam software, maka harus dapat menggeser significand bit dalam beberapa operasi
    - Untuk mengubah dari IRA ke packed desimal, kita perlu mengekstrak 4 bit paling kanan dari setiap byte

Tipe Data x86

Data Type	Description
General	Byte, word (16 bits), doubleword (32 bits), quadword (64 bits), and double quadword (128 bits) locations with arbitrary binary contents.
Integer	A signed binary value contained in a byte, word, or doubleword, using twos complement representation.
Ordinal	An unsigned integer contained in a byte, word, or doubleword.
Unpacked binary coded decimal (BCD)	A representation of a BCD digit in the range 0 through 9, with one digit in each byte.
Packed BCD	Packed byte representation of two BCD digits; value in the range 0 to 99.
Near pointer	A 16-bit, 32-bit, or 64-bit effective address that represents the offset within a segment. Used for all pointers in a nonsegmented memory and for references within a segment in a segmented memory.
Far pointer	A logical address consisting of a 16-bit segment selector and an offset of 16, 32, or 64 bits. Far pointers are used for memory references in a segmented memory model where the identity of a segment being accessed must be specified explicitly.
Bit field	A contiguous sequence of bits in which the position of each bit is considered as an independent unit. A bit string can begin at any bit position of any byte and can contain up to 32 bits.
Bit string	A contiguous sequence of bits, containing from zero to $2^{23} - 1$ bits.
Byte string	A contiguous sequence of bytes, words, or doublewords, containing from zero to $2^{23} - 1$ bytes.
Floating point	See Figure 12.4.
Packed SIMD (single instruction, multiple data)	Packed 64-bit and 128-bit data types.

Format Data Numerik x86



Tipe Data Single-Instruction-Multiple-Data (SIMD)

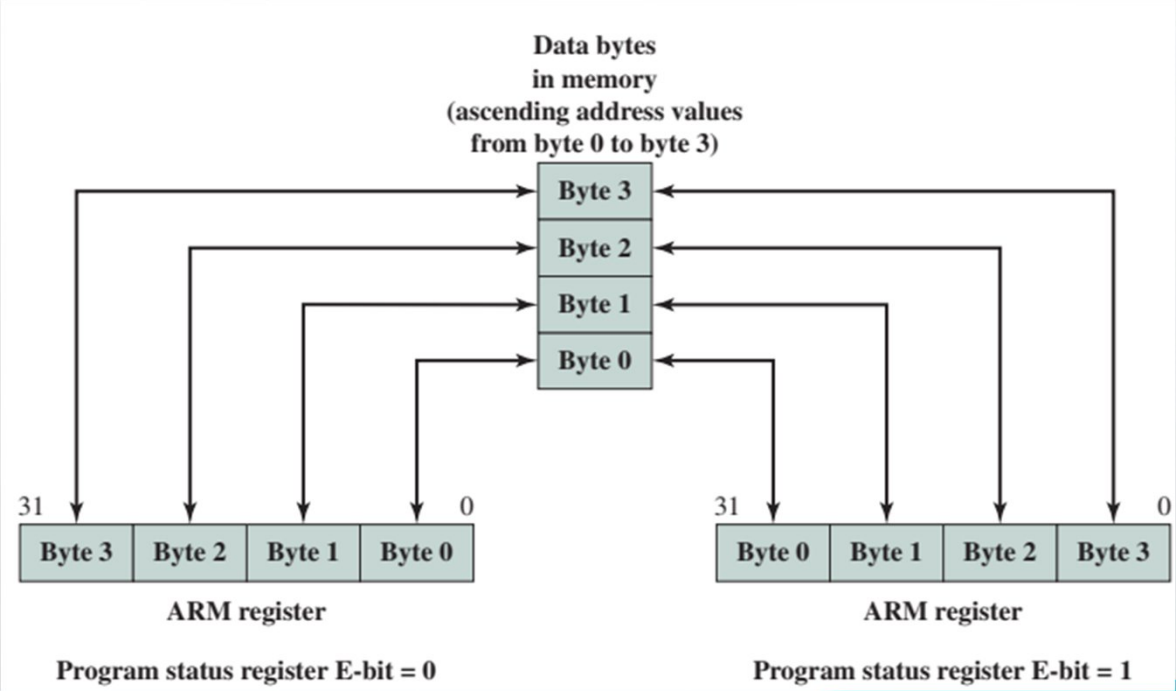
- **Diperkenalkan pada arsitektur x86 sebagai bagian dari ekstensi set instruksi untuk mengoptimalkan kinerja aplikasi multimedia**
- **Ekstensi ini termasuk MMX (multimedia extensions) dan SSE (streaming SIMD extensions)**
- **Tipe data:**
  - Packed byte dan packed byte integer
  - Packed word dan packed word integer
  - Packed doubleword dan packed doubleword integer
  - Packed quadword dan packed quadword integer
  - Packed single-precision floating-point dan packed double-precision floating-point

Tipe Data ARM

- **Tipe data panjang 8 (byte), 16 (halfword), 32 (word) bit.**
- **Biasanya, akses halfword harus selaras dan akses word harus selaras pula**
- **3 alternatif akses tidak selaras:**
  - Default case
    - Alamat terpotong, dengan bit alamat [1: 0] diperlakukan sebagai zero untuk akses word, dan bit alamat [0] diperlakukan sebagai zero untuk akses halfword
    - Muat satu instruksi word ARM didefinisikan secara arsitektural untuk memutar data word selaras yang ditransfer oleh alamat bukan word selaras satu, dua, atau tiga byte tergantung pada nilai dari dua alamat least significant bit
  - Alignment checking
    - Bila bit kontrol yang sesuai diatur, sinyal data abort menunjukkan kesalahan penyelarasan untuk akses yang tidak selaras
  - Unaligned access
    - Bila opsi ini diaktifkan, prosesor menggunakan satu atau lebih akses memori untuk menghasilkan transfer byte yang berdekatan secara transparan ke programmer

ARM Endian

- State bit (E-bit) pada register kontrol, diatur dan dihapus di bawah program kontrol menggunakan instruksi SETEND
- E-bit menentukan endian mana yang memuat dan menyimpan data



Operasi Set Instruksi Umum

Type	Operation Name	Description
Data transfer	Move (transfer)	Transfer word or block from source to destination
	Store	Transfer word from processor to memory
	Load (fetch)	Transfer word from memory to processor
	Exchange	Swap contents of source and destination
	Clear (reset)	Transfer word of 0s to destination
	Set	Transfer word of 1s to destination
	Push	Transfer word from source to top of stack
	Pop	Transfer word from top of stack to destination
Arithmetic	Add	Compute sum of two operands
	Subtract	Compute difference of two operands
	Multiply	Compute product of two operands
	Divide	Compute quotient of two operands
	Absolute	Replace operand by its absolute value
	Negate	Change sign of operand
	Increment	Add 1 to operand
	Decrement	Subtract 1 from operand
Logical	AND	Perform logical AND
	OR	Perform logical OR
	NOT	(complement) Perform logical NOT
	Exclusive-OR	Perform logical XOR
	Test	Test specified condition; set flag(s) based on outcome
	Compare	Make logical or arithmetic comparison of two or more operands; set flag(s) based on outcome
	Set Control Variables	Class of instructions to set controls for protection purposes, interrupt handling, timer control, etc.
	Shift	Left (right) shift operand, introducing constants at end
	Rotate	Left (right) shift operand, with wraparound end
Transfer of control	Jump (branch)	Unconditional transfer; load PC with specified address
	Jump Conditional	Test specified condition; either load PC with specified address or do nothing, based on condition
	Jump to Subroutine	Place current program control information in known location; jump to specified address
	Return	Replace contents of PC and other register from known location
	Execute	Fetch operand from specified location and execute as instruction; do not modify PC
	Skip	Increment PC to skip next instruction
	Skip Conditional	Test specified condition; either skip or do nothing based on condition
	Halt	Stop program execution
	Wait (hold)	Stop program execution; test specified condition repeatedly; resume execution when condition is satisfied
	No operation	No operation is performed, but program execution is continued
Input/output	Input (read)	Transfer data from specified I/O port or device to destination (e.g., main memory or processor register)
	Output (write)	Transfer data from specified source to I/O port or device
	Start I/O	Transfer instructions to I/O processor to initiate I/O operation
	Test I/O	Transfer status information from I/O system to specified destination
Conversion	Translate	Translate values in a section of memory based on a table of correspondences
	Convert	Convert the contents of a word from one form to another (e.g., packed decimal to binary)

Aksi Prosesor untuk Jenis Operasi

Data transfer	Transfer data from one location to another
	If memory is involved: Determine memory address Perform virtual-to-actual-memory address transformation Check cache Initiate memory read/write
Arithmetic	May involve data transfer, before and/or after
	Perform function in ALU
	Set condition codes and flags
Logical	Same as arithmetic
Conversion	Similar to arithmetic and logical. May involve special logic to perform conversion
Transfer of control	Update program counter. For subroutine call/return, manage parameter passing and linkage
I/O	Issue command to I/O module
	If memory-mapped I/O, determine memory-mapped address

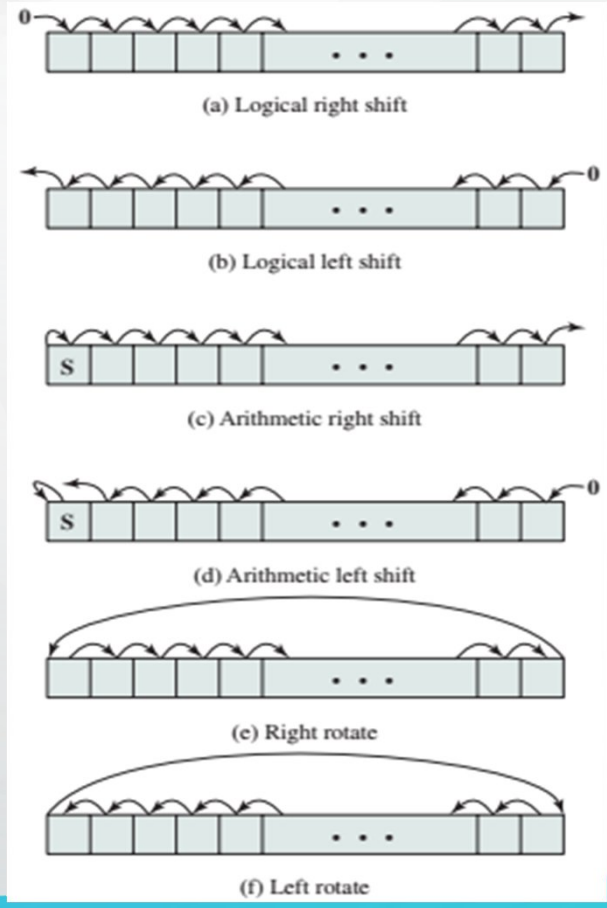
Transfer Data

- **Jenis paling dasar dari instruksi mesin adalah instruksi transfer data**
- **Harus ditentukan:**
  - lokasi operand sumber dan tujuan harus ditentukan
  - panjang data yang akan ditransfer harus ditunjukkan
  - seperti semua instruksi dengan operan, mode pengalamatan untuk setiap operand harus ditentukan
- **Jika sumber dan tujuan adalah register, maka prosesor menyebabkan data dipindahkan dari satu register ke register lain → operasi internal prosesor.**
- **Jika satu atau kedua operan ada dalam memori, maka prosesor harus melakukan beberapa atau semua tindakan berikut:**
  - Hitung alamat memori berdasarkan mode pengalamatan
  - Jika alamat mengacu pada memori virtual, terjemahkan dari alamat memori virtual ke memori nyata
  - Tentukan apakah item yang dialamatkan ada di cache
  - Jika tidak, berikan perintah ke modul memori

Aritmatika dan Logika

- Kebanyakan mesin menyediakan operasi aritmatika dasar untuk menambah, mengurangi, mengalikan, dan membagi
- Disediakan untuk bilangan integer fixed point
- Seringkali untuk bilangan floating-point dan bilangan desimal

Input	Operation	Result
10100110	Logical right shift (3 bits)	00010100
10100110	Logical left shift (3 bits)	00110000
10100110	Arithmetic right shift (3 bits)	11110100
10100110	Arithmetic left shift (3 bits)	10110000
10100110	Right rotate (3 bits)	11010100
10100110	Left rotate (3 bits)	00110101



Konversi

- **Instruksi yang mengubah format atau mengoperasikan format data**
- **Contoh → konversi dari desimal ke biner**
- **Contoh instruksi pengeditan yang lebih kompleks adalah instruksi EAS / 390 Translate (TR)**
  - Konversi dari satu kode 8-bit ke yang lain, dan dibutuhkan tiga operan:

    TR R1 (L), R2
  - R2 berisi alamat awal tabel kode 8-bit
  - Byte L yang dimulai dari alamat yang ditentukan di R1 diterjemahkan, setiap byte diganti dengan konten entri tabel yang diindeks oleh byte itu

Input / Output

- **Terdapat beberapa pendekatan:**
  - Isolated programmed I/O
  - Memory-mapped programmed I/O
  - Direct Memory Access (DMA)
  - Penggunaan prosesor I/O
- **Banyak implementasi hanya menyediakan sedikit instruksi I / O, dengan tindakan spesifik yang ditentukan oleh parameter, kode, atau command word**

Kontrol Sistem

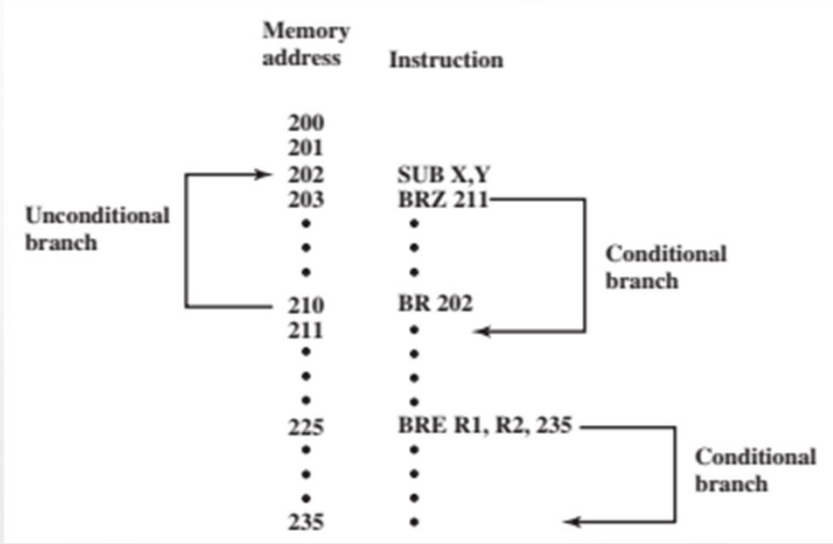
- **Instruksi yang dapat dijalankan hanya saat prosesor berada dalam privileged tertentu atau menjalankan program di area memori khusus**
- **Biasanya instruksi ini disediakan untuk penggunaan sistem operasi**
- **Contoh operasi kontrol sistem:**
  - Dapat membaca atau mengubah register kontrol
  - Instruksi untuk membaca atau memodifikasi kunci perlindungan penyimpanan
  - Akses ke Process Control Block dalam sistem multiprogramming

Transfer Kontrol

- **Operasi transfer kontrol diperlukan karena:**
  - Penting untuk dapat menjalankan setiap instruksi lebih dari satu kali
  - Hampir semua program melibatkan beberapa pengambilan keputusan
  - Akan membantu jika ada mekanisme untuk memecah task menjadi bagian-bagian kecil yang dapat dikerjakan satu per satu2
- **Beberapa operasi transfer kontrol umum yang ada di set instruksi:**
  - Branch
  - Skip
  - Procedure call

Branch

- **Branch = jump**
  - salah satu operannya memiliki alamat dari instruksi selanjutnya yang akan dieksekusi
- **Conditional branch**
  - perintah – perintah eksekusi percabangan yang memerlukan syarat tertentu agar dihasilkan suatu nilai dari percabangan tersebut
- **Unconditional branch**
  - perintah – perintah eksekusi percabangan tanpa syarat tertentu



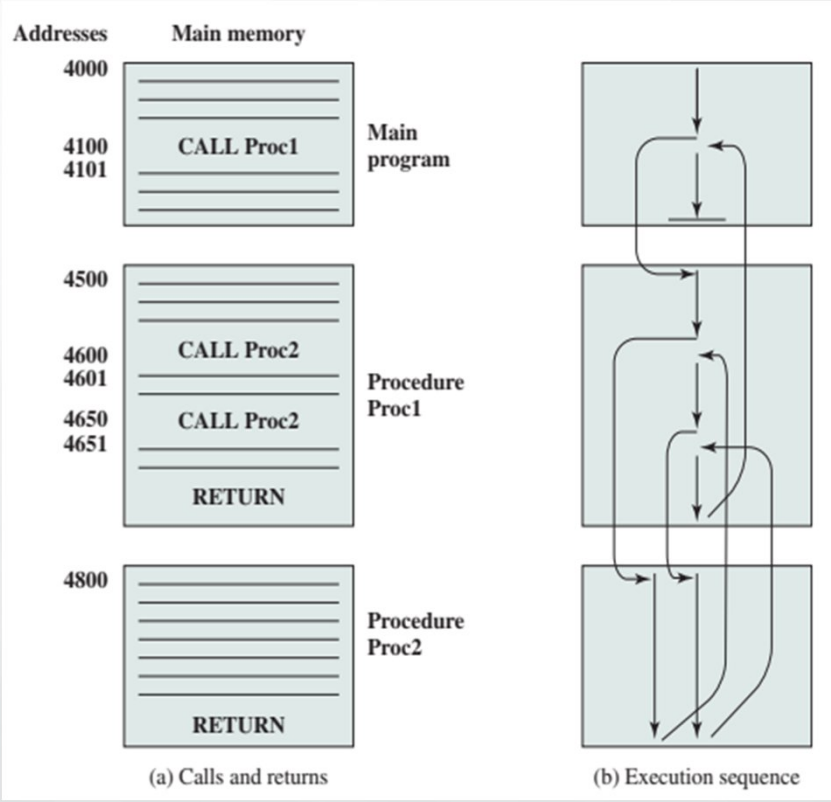
Skip

- Biasanya menyiratkan bahwa satu instruksi dilewati, sehingga alamat tersirat sama dengan alamat instruksi berikutnya ditambah satu panjang instruksi
- Karena instruksi skip tidak memerlukan field alamat tujuan, maka bebas melakukan hal-hal lain
- Contoh instruksi increment-and-skip-if-zero (ISZ)

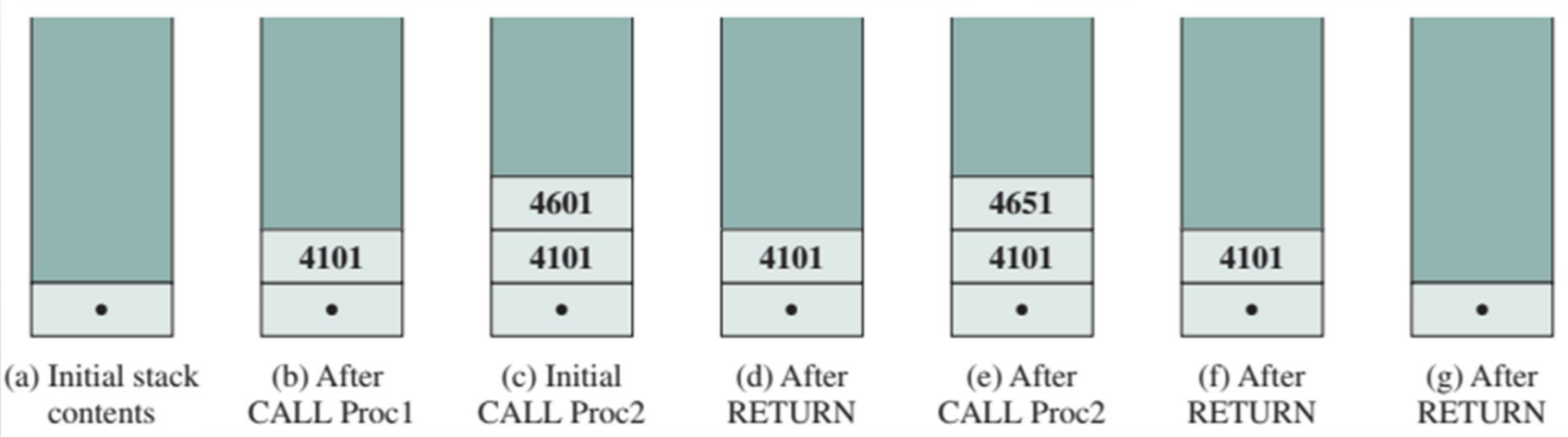
```
301
 :
309   ISZ   R1
310   BR    301
311
```

Procedure Call

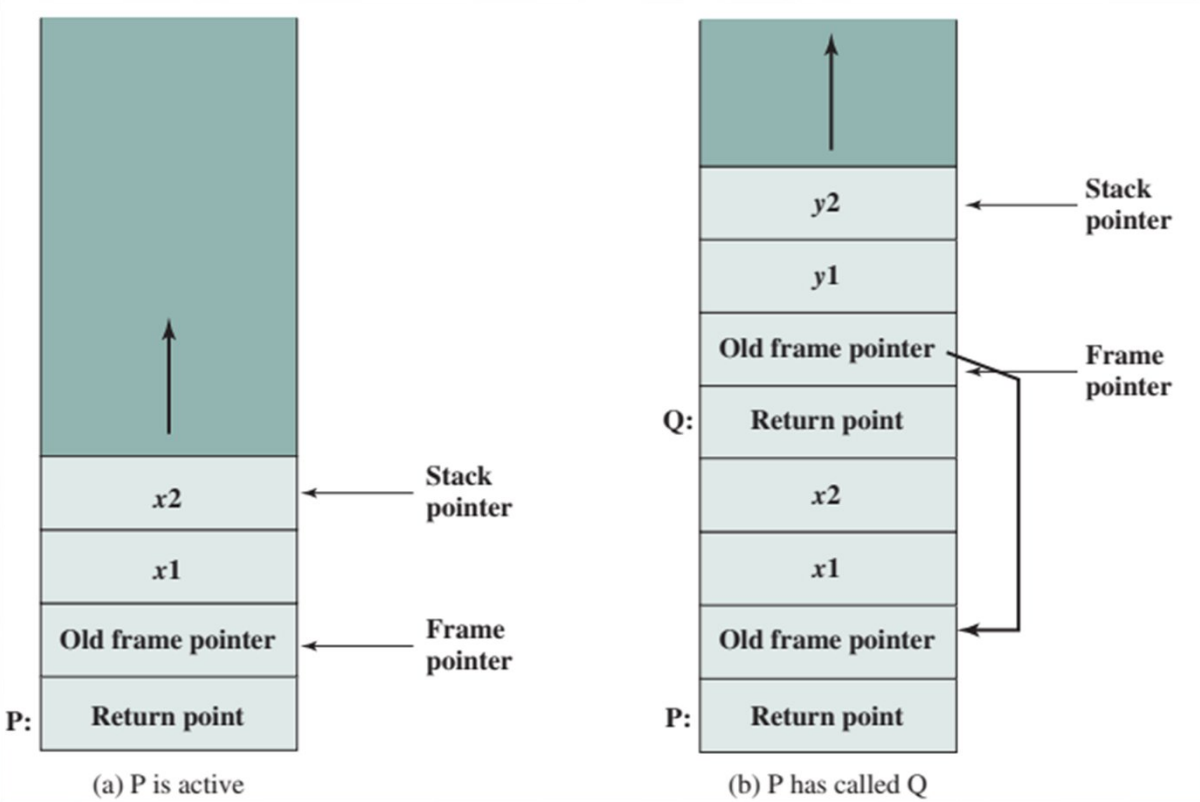
- **Program komputer mandiri yang digabungkan ke dalam program yang lebih besar**
  - Pada titik manapun dalam program, prosedur dapat dipanggil
  - Prosesor diinstruksikan untuk pergi dan menjalankan seluruh prosedur dan kemudian kembali ke titik di mana panggilan itu dilakukan
- **Alasan utama penggunaan prosedur:**
  - Ekonomis
    - Prosedur memungkinkan potongan kode yang sama digunakan berkali-kali
  - Modularitas (terpisah)
- **Melibatkan 2 instruksi dasar**
  - Instruksi call yang bercabang dari lokasi sekarang ke prosedur
  - Instruksi return yang mengembalikan dari prosedur ke tempat pemanggilannya



Penggunaan Stack



Stack Frame


x2
x1
Old frame pointer
Return point
P:
Stack pointer
Frame pointer
(a) P is active
y2
y1
Old frame pointer
Return point
x2
x1
Old frame pointer
Return point
Q:
P:
Stack pointer
Frame pointer
(b) P has called Q

Jenis Operasi Intel x86 dan AMD

- **x86 menyediakan tipe operasi array yang kompleks termasuk sejumlah instruksi khusus**
- **Tujuan → menyediakan tools bagi compiler untuk menghasilkan terjemahan bahasa mesin yang dioptimalkan untuk program bahasa tingkat tinggi**
- **Menyediakan 4 instruksi untuk mendukung procedure call/return:**
  - CALL
  - ENTER
  - LEAVE
  - RETURN
- **Ketika prosedur baru dipanggil, hal berikut harus dilakukan saat masuk ke prosedur baru:**
  - Push return point pada stack
  - Push current frame pada stack
  - Salin stack pointer sebagai nilai baru frame pointer
  - Sesuaikan stack pointer untuk mengalokasikan frame

x86 Status Flags dan Condition Code

- Status flags → bit dalam register khusus yang dapat diatur oleh operasi tertentu dan digunakan dalam instruksi conditional branch
- Condition code → mengacu pada pengaturan dari satu atau lebih status flags
- Di x86 dan banyak arsitektur lainnya, tanda status ditetapkan oleh operasi aritmatika dan perbandingan.
- Operasi perbandingan di sebagian besar bahasa mengurangi dua operand, seperti halnya operasi pengurangan

x86 Status Flags

Status Bit	Name	Description
C	Carry	Indicates carrying or borrowing out of the left-most bit position following an arithmetic operation. Also modified by some of the shift and rotate operations.
P	Parity	Parity of the least-significant byte of the result of an arithmetic or logic operation. 1 indicates even parity; 0 indicates odd parity.
A	Auxiliary Carry	Represents carrying or borrowing between half-bytes of an 8-bit arithmetic or logic operation. Used in binary-coded decimal arithmetic.
Z	Zero	Indicates that the result of an arithmetic or logic operation is 0.
S	Sign	Indicates the sign of the result of an arithmetic or logic operation.
O	Overflow	Indicates an arithmetic overflow after an addition or subtraction for twos complement arithmetic.

x86 Condition Code

Symbol	Condition Tested	Comment
A, NBE	C = 0 AND Z = 0	Above; Not below or equal (greater than, unsigned)
AE, NB, NC	C = 0	Above or equal; Not below (greater than or equal, unsigned); Not carry
B, NAE, C	C = 1	Below; Not above or equal (less than, unsigned); Carry set
BE, NA	C = 1 OR Z = 1	Below or equal; Not above (less than or equal, unsigned)
E, Z	Z = 1	Equal; Zero (signed or unsigned)
G, NLE	[(S = 1 AND O = 1) OR (S = 0 AND O = 0)]AND[Z = 0]	Greater than; Not less than or equal (signed)
GE, NL	(S = 1 AND O = 1) OR (S = 0 AND O = 0)	Greater than or equal; Not less than (signed)
L, NGE	(S = 1 AND O = 0) OR (S = 0 AND O = 0)	Less than; Not greater than or equal (signed)
LE, NG	(S = 1 AND O = 0) OR (S = 0 AND O = 1) OR (Z = 1)	Less than or equal; Not greater than (signed)
NE, NZ	Z = 0	Not equal; Not zero (signed or unsigned)
NO	O = 0	No overflow
NS	S = 0	Not sign (not negative)
NP, PO	P = 0	Not parity; Parity odd
O	O = 1	Overflow
P	P = 1	Parity; Parity even
S	S = 1	Sign (negative)

Instruksi x86 SIMD

- **Pada tahun 1996, Intel memperkenalkan teknologi MMX ke dalam produk Pentiumnya**
  - sekumpulan instruksi yang sangat dioptimalkan untuk tugas-tugas multimedia
- **Data video dan audio biasanya terdiri dari array besar dengan tipe data kecil**
- **3 tipe data baru di MMX:**
  - Packet byte
  - Packet word
  - Packet doubleword
- **Setiap tipe data memiliki panjang 64 bit dan terdiri dari beberapa field data yang lebih kecil, masing-masing menggunakan fixed-point integer**

Set Instruksi MMX

Category	Instruction	Description
Arithmetic	PADD [B, W, D]	Parallel add of packed eight bytes, four 16-bit words, or two 32-bit doublewords, with wraparound.
	PADDS [B, W]	Add with saturation.
	PADDUS [B, W]	Add unsigned with saturation.
	PSUB [B, W, D]	Subtract with wraparound.
	PSUBS [B, W]	Subtract with saturation.
	PSUBUS [B, W]	Subtract unsigned with saturation.
	PMULHW	Parallel multiply of four signed 16-bit words, with high-order 16 bits of 32-bit result chosen.
	PMULLW	Parallel multiply of four signed 16-bit words, with low-order 16 bits of 32-bit result chosen.
	PMADDWD	Parallel multiply of four signed 16-bit words; add together adjacent pairs of 32-bit results.
Comparison	PCMPEQ [B, W, D]	Parallel compare for equality; result is mask of 1s if true or 0s if false.
	PCMPGT [B, W, D]	Parallel compare for greater than; result is mask of 1s if true or 0s if false.
Conversion	PACKUSWB	Pack words into bytes with unsigned saturation.
	PACKSS [WB, DW]	Pack words into bytes, or doublewords into words, with signed saturation.
	PUNPCKH [BW, WD, DQ]	Parallel unpack (interleaved merge) high-order bytes, words, or doublewords from MMX register.
	PUNPCKL [BW, WD, DQ]	Parallel unpack (interleaved merge) low-order bytes, words, or doublewords from MMX register.
Logical	PAND	64-bit bitwise logical AND
	PNDN	64-bit bitwise logical AND NOT
	POR	64-bit bitwise logical OR
	PXOR	64-bit bitwise logical XOR
Shift	PSLL [W, D, Q]	Parallel logical left shift of packed words, doublewords, or quadword by amount specified in MMX register or immediate value.
	PSRL [W, D, Q]	Parallel logical right shift of packed words, doublewords, or quadword.
	PSRA [W, D]	Parallel arithmetic right shift of packed words, doublewords, or quadword.
Data transfer	MOV [D, Q]	Move doubleword or quadword to/from MMX register.
Statemgt	EMMS	Empty MMX state (empty FP registers tag bits).

Jenis Operasi ARM

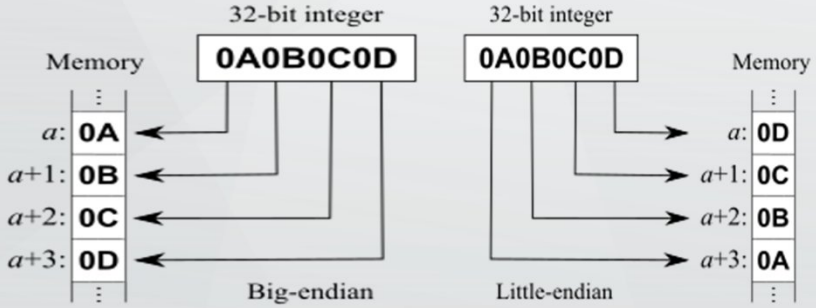
- **ARM menyediakan banyak jenis operasi:**
  - Load and store
  - Branch
  - Data-processing
  - Multiply
  - Parallel addition and subtraction
  - Extend
  - Status register access

ARM Conditional Instruction Execution

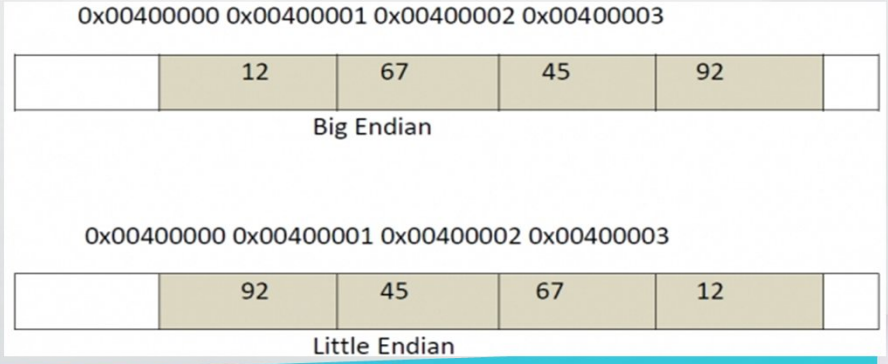
Code	Symbol	Condition Tested	Comment
0000	EQ	Z = 1	Equal
0001	NE	Z = 0	Not equal
0010	CS/HS	C = 1	Carry set/unsigned higher or same
0011	CC/LO	C = 0	Carry clear/unsigned lower
0100	MI	N = 1	Minus/negative
0101	PL	N = 0	Plus/positive or zero
0110	VS	V = 1	Overflow
0111	VC	V = 0	No overflow
1000	HI	C = 1 AND Z = 0	Unsigned higher
1001	LS	C = 0 OR Z = 1	Unsigned lower or same
1010	GE	N = V [(N = 1 AND V = 1) OR (N = 0 AND V = 0)]	Signed greater than or equal
1011	LT	N ≠ V [(N = 1 AND V = 0) OR (N = 0 AND V = 1)]	Signed less than
1100	GT	(Z = 0) AND (N = V)	Signed greater than
1101	LE	(Z = 1) OR (N ≠ V)	Signed less than or equal
1110	AL	–	Always (unconditional)
1111	–	–	This instruction can only be executed unconditionally

Big-Endian dan Little-Endian

- Endiannes → bagaimana byte diatur dalam penyimpanan memori
- Ada 2 jenis, yaitu Big-Endian dan Little-Endian
- Big-Endian → "Big-End" atau nilai dari Most Significant Byte (MSB) disimpan pertama pada urutan byte (alamat terendah). Sisa data ditempatkan secara berurutan dalam tiga byte berikutnya dalam memori.
- Little-Endian → "Little-End" atau nilai dari Least Significant Byte (LSB) disimpan pertama pada urutan byte (alamat terendah). Sisa data ditempatkan secara berurutan dalam tiga byte berikutnya dalam memori.



0x12674592 → 32 bit



TERIMA KASIH