PROGRAM STUDI

S1 SISTEM KOMPUTER

UNIVERSITAS DIPONEGORO

ORGANISASI KOMPUTER

Instruksi Mesin & Program

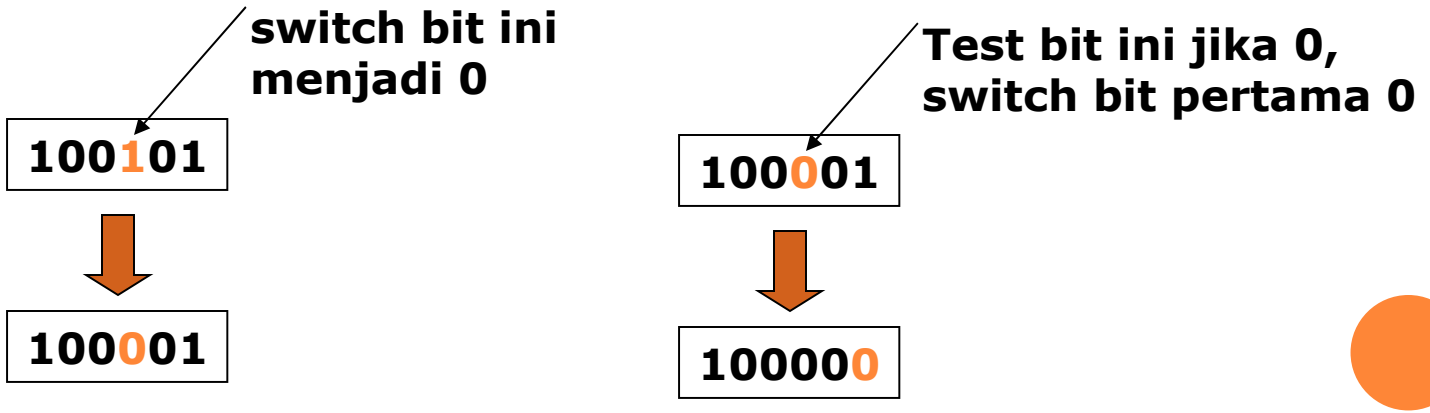
Oky Dwi Nurhayati, ST, MT
email: okydn@undip.ac.id

TUJUAN INSTRUKSIONAL

- Mahasiswa mengenal instruksi mesin dan eksekusi program
- Metode pengalamatan untuk mengakses register dan operand memori
- Bahasa Assembly untuk merepresentasikan instruksi mesin,data dan program

KOMPUTER DIGITAL

- Komputer Digital:
  - Hanya mengenal dua status (mis. ada / tidak ada tegangan)
    - Sangat sederhana hanya dapat bernilai: 1 atau 0 → biner
  - Operasi hanya dapat dilakukan pada bit; yang dapat bernilai 1 atau 0.
    - Contoh operasi mengubah (*flip*, *switch*) nilai bit, menjadikan bit tertentu 0; test bit jika 0 atau bukan.



KOMPUTER & "*THINGS*"

- … jadi, apakah komputer (yang hanya mengenal bits dengan nilai: 0 dan 1) demikian "powerful"?

- Apakah komputer dapat merepresentasikan "sesuatu", apa saja?

simulasi mobil

..hello buzz

BIT

- Dengan bit, bagaimana komputer dapat merepresentasikan:
  - Bilangan (numerik)? Alfabet ? Kata? Alamat? Gambar?
  - Contoh: Bilangan
    - Manusia lebih mudah menggunakan representasi/ notasi desimal.
    - Misalkan: 1, 25, 125, 3896754321
    - Disebut basis 10, dengan simbol:
      Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- ..tapi komputer hanya mengenal 2 simbol (0 dan 1)

  Bagaimana melakukan representasi bilangan yang dikenal manusia?

CONTOH: BILANGAN!

- Bilangan Basis B → B simbol per digit:
  - Basis 10 (Decimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    Basis 2 (Binary): 0, 1
- Komputer menyimpan dan beroperasi dalam "binary" → Basis 2
  - Dapat melakukan konversi (representasi) bilangan dari basis 10 ke basis 2 (dan sebaliknya).
  - Decimal: 0,1,2,3,4,5,6,7,8,9

    $90 = 9x10^1 + 0x10^0$
  - Binary: 0,1

    $1011010 = 1x2^6 + 0x2^5 + 1x2^4 + 1x2^3 + 0x2^2 + 1x2 + 0x2^0$

    $= 64 + 16 + 8 + 2 = 90$

BIT DAPAT MEPRESENTASIKAN "APA SAJA" !!!

- Bits dapat merepresentasikan apapun!
  - Karakter? Latin:
    - 26 huruf => 5 bits
    - Huruf besar/kecil + tanda lain => 7 bits, berapa simbol huruf?
    - Karakter, bahasa lain => 16 (unicode)
  - Logical values?
    - 0 -> False, 1 => True
  - Warna ? Berapa banyak warna => berapa bits?
  - Alamat? (berapa karakter alfabet ..)

- .. Tapi N bits → hanya dapat merepresentasikan 2^N sesuatu

APA LAGI (SESUATU YANG BERGUNA) YANG DAPAT DILAKUKAN?

- Apa yang dapat dilakukan dengan "bilangan"?
  - … umumnya semua operasi (instruksi) yang biasa dilakukan pada bilangan!
    - Tambahkan, Kurangkan, Kalikan, Bagikan, Bandingkan!
  - Contoh: 10 + 7 = 17

Apakah instruksi +, dapat diwakili oleh bits?

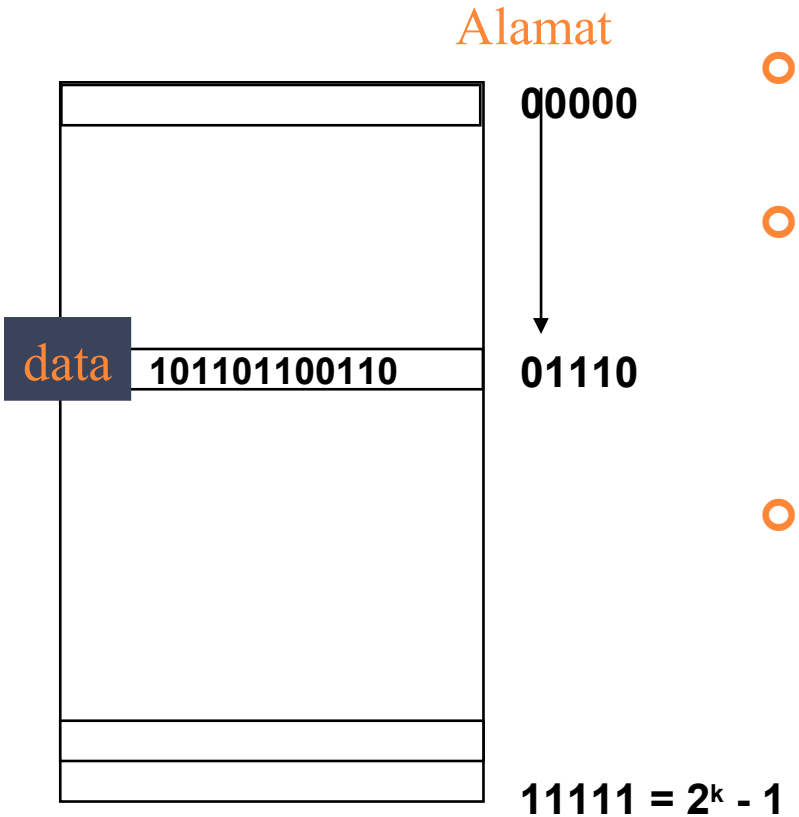
```
  1  1
  1  0  1  0      → 10
+ 0  1  1  1      → 7
-------------
1  0  0  0  1     → 17
```

BIT → INSTRUKSI

- Instruksi (Operasi). Apakah dapat diwakili oleh bit?
  - Contoh:
    - 0 => tepuk tangan
    - 1 => snap jari jempol dan telunjuk
    - Eksekusi Instruksi: 1 0 1 1 0 0
- Jadi instruksi operasi bilangan (sebagai contoh!)
  - Misalkan 3 bit (berapa banyak instruksi?):
    - 000 => tambahkan
    - 001 => kurangkan
    - 010 => kalikan
    - 011 => bagikan
    - 100 => bandingkan dst.
- Jadi bit (data) dapat diartikan sebagai instruksi!

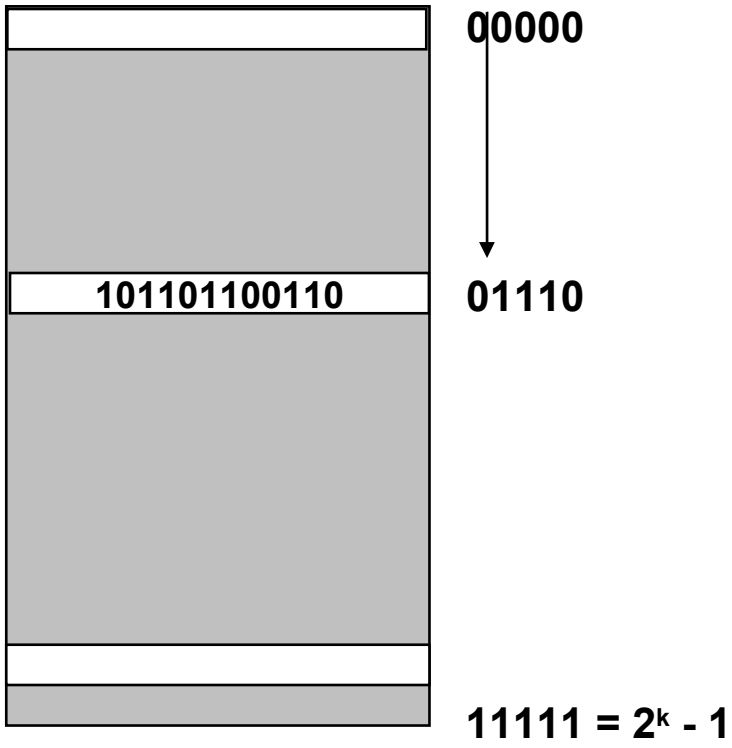
Memori

BILANGAN (DATA) DISIMPAN DI MEMORI



- Memori adalah tempat menyimpan bit data
- Suatu "word" adalah sejumlah bit data tetap, (mis. 16, atau 32 bit) pada satu lokasi di memori
- Alamat menunjuk ke lokasi "word" disimpan.
  - Alamat dapat direpresen-tasikan oleh bit
  - Alamat juga sebagai "bilangan" (yang dapat dimanipulasikan)

APA SAJA YANG DAPAT DISIMPAN?



- Apa yang dapat disimpan?
  - Bilangan
  - Karakter
  - Alamat data
  - Representasi “sesuatu” di dunia luar
  - ..

anything

Big Idea: Komputer dapat menyimpan apapun.

PENGKODEAN INFORMASI: REPRESENTASI DATA

- Binary: 0,1

  $1011010 = 1x2^6 + 0x2^5 + 1x2^4 + 1x2^3 + 0x2^2 + 1x2 + 0x2^0$

  $= 64 + 16 + 8 + 2 = 90$

- Hexa-Decimal: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

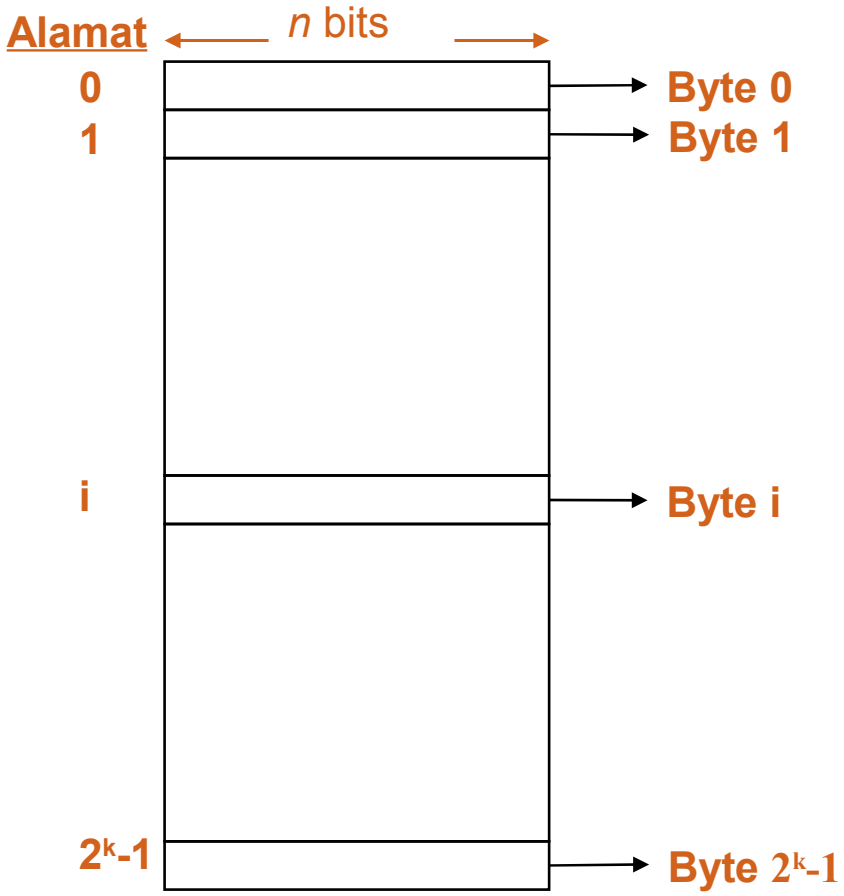
  $5A = 5x16^1 + 10x16^0$

  $= 80 + 10 = 90$

  - Penulisan: 0x5A

- Bit String:
  - 4 bit → nibble
  - 8 bit → byte
  - 16 bit → half-word
  - 32 bit → word
  - 64 bit → double-word

PENGALAMATAN MEMORI



- *k* menentukan besarnya ruang alamat (*address space*) memori:
  - k = 16 → ruang alamat = 2^{16} (64536) lokasi
  - k = 32 → ruang alamat = 2^{32} (4 G) lokasi
- *n* menentukan besarnya suatu *word* (jumlah bit)
  - *n* = 8, 16, 32, 64
- Umumnya ukuran peng-alamatan terkecil adalah dalam orde byte → *byte addressable*

PENGALAMATAN OBJEK: *ENDIANESS*

➲ cara menata bagian-bagian dari suatu objek (yang berukuran > 1 byte) di memori

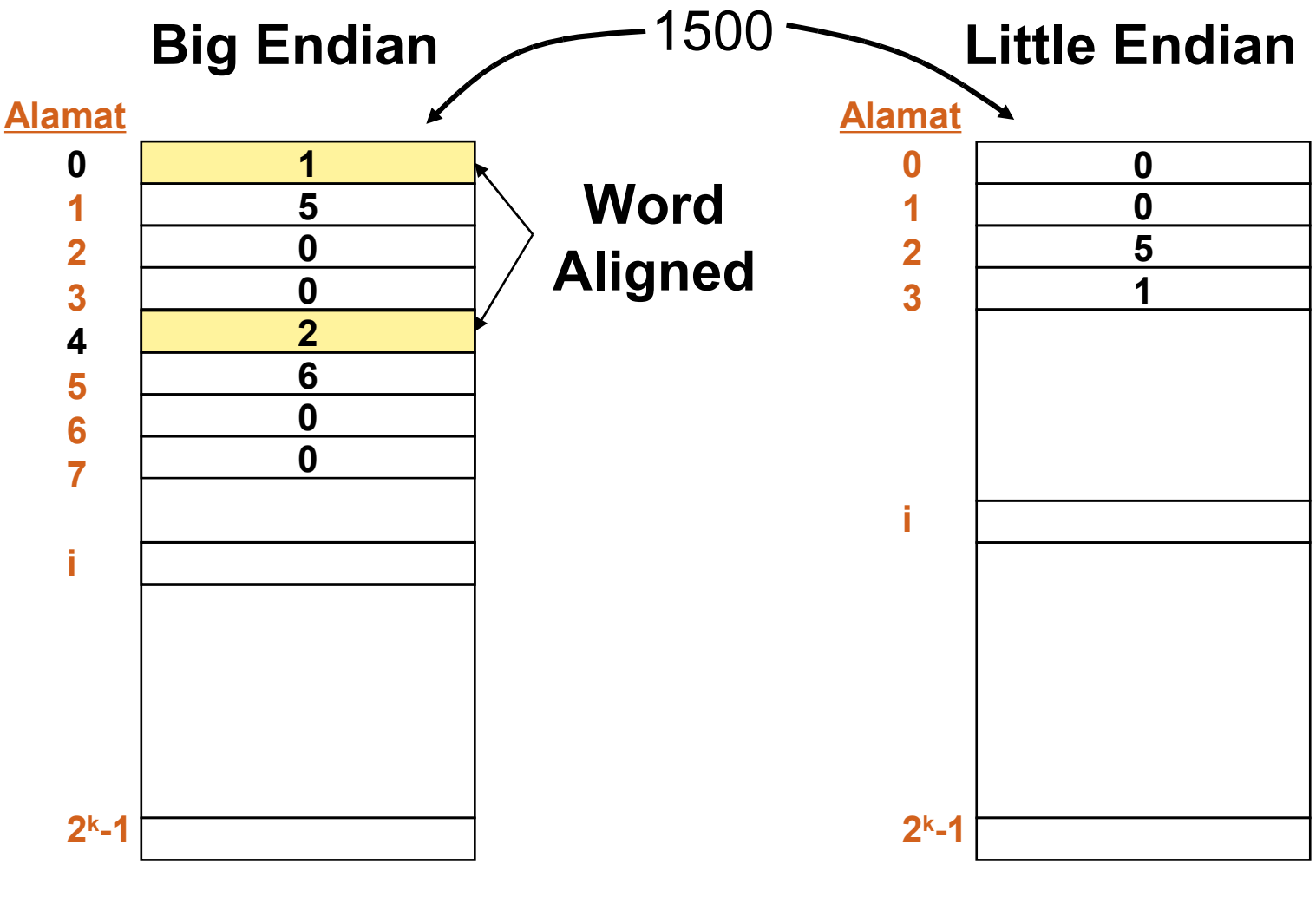
- Big Endian: address of most significant
  - IBM 360/370, Motorola 68k, MIPS, Sparc, HP PA

0	msb			lsb
1	1	5	0	0

- Little Endian: address of least significant
  - Intel 80x86, DEC Vax, DEC Alpha (Windows NT)

0	lsb			msb
1	0	0	5	1

ENDIANESS & WORD ALIGNMENT



Stored Program

THE STORED PROGRAM COMPUTER (1/2)

- Konsep Kunci:

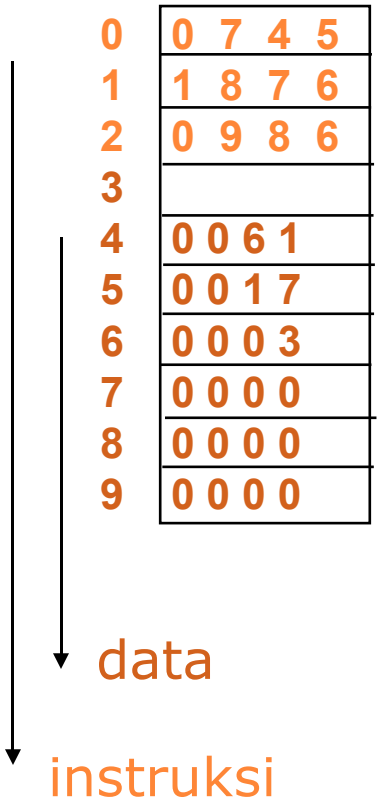
  Data dapat diartikan sebagai instruksi!

  - Data di komputer mampu merepresentasikan sesuatu (*thing, anything*)! Tergantung intrepetasi dan operasi yang diinginkan.

- BIG IDEA: STORED PROGRAM → program dapat disimpan sebagai data dan dijalankan oleh komputer

  - Merupakan konsep awal komputer → Von Neumann Architecture (1955).

THE STORED PROGRAM COMPUTER (2/2)

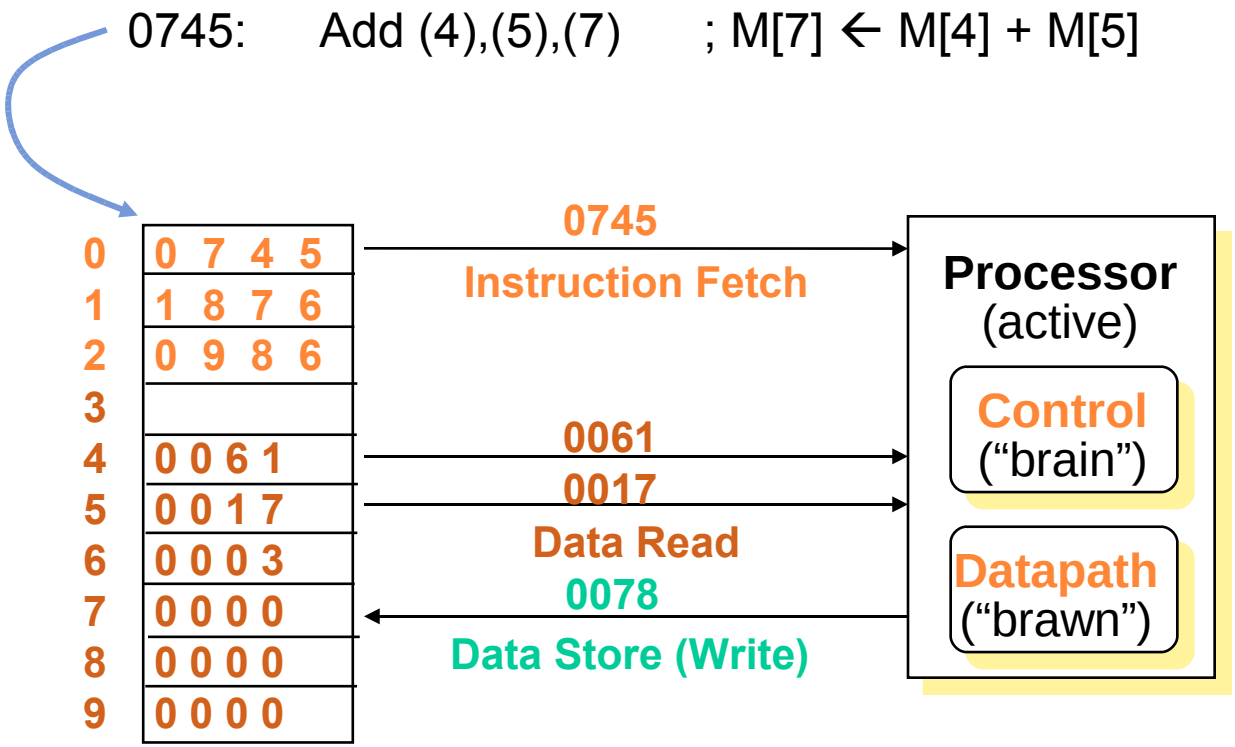


- Memori menyimpan instruksi dan data sebagai bit.
- Instruksi diambil oleh prosesor dari memori, diartikan, dan, dieksekusi (*operands*/data diambil, diolah, dan disimpan ke memori).
- Contoh Instruksi 4-digit
  - Operasi: 0 => add, 1 => sub
  - Alamat hasil
  - Alamat op1
  - Alamat op2

instruksi 0: 0745
0=add (jenis instruksi),
7=addr. result,
4=addr op1,
5=addr op2

Apa yang berada di lokasi 9 setelah eksekusi instruksi 0, 1, 2?

OPERASI PADA MEMORI



JADI, APA ARTINYA?

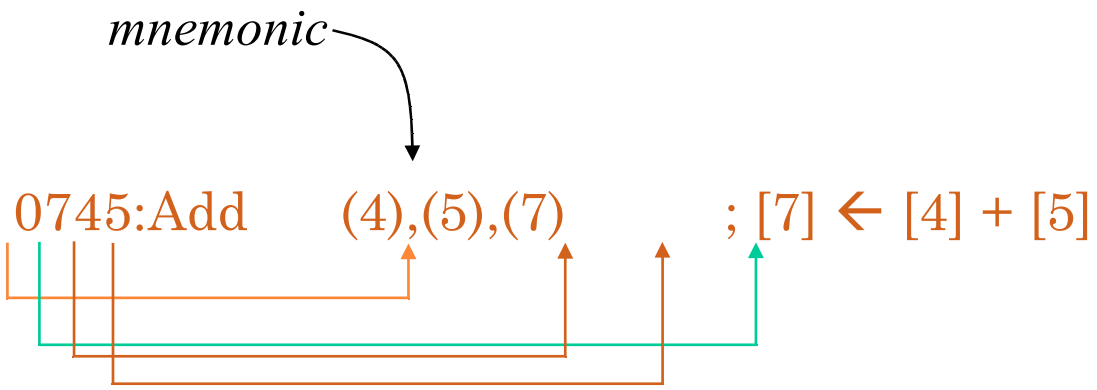
- Kita dapat menulis sebuah program yang dapat "menerjemahkan" untaian karakter ke "instruksi komputer (bit)".
  - Program tersebut disebut: *compiler* atau *assembler*.
- Kita dapat me-*load* hasil pola bit ke memori dan dijalankan oleh prosesor:
  - mampu mengolah/manipulasi bilangan, karakter, *pixels* … (Aplikasi, *Software*)
  - mampu melakukan penerjemahan perintah ke instruksi komputer (*Compiler*)
  - dapat mengontrol komputer → *load* dan jalankan program (*Operating Systems*)

Set Instruksi

SET INSTRUKSI (BAHASA MESIN) ↔ BAHASA RAKITAN

- **Bahasa Mesin → kumpulan bit yang merepresentasikan Operasi & Operand**

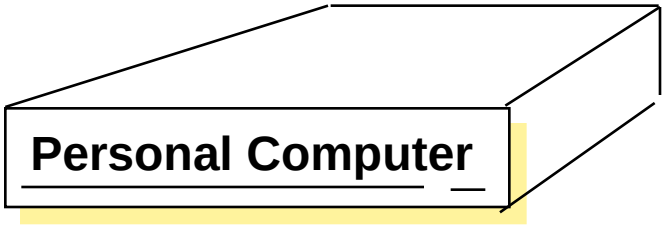
- **Bahasa Rakitan → representasi dari Bahasa Mesin dalam bahasa (kumpulan huruf & angka) yang lebih mudah dimengerti oleh manusia**



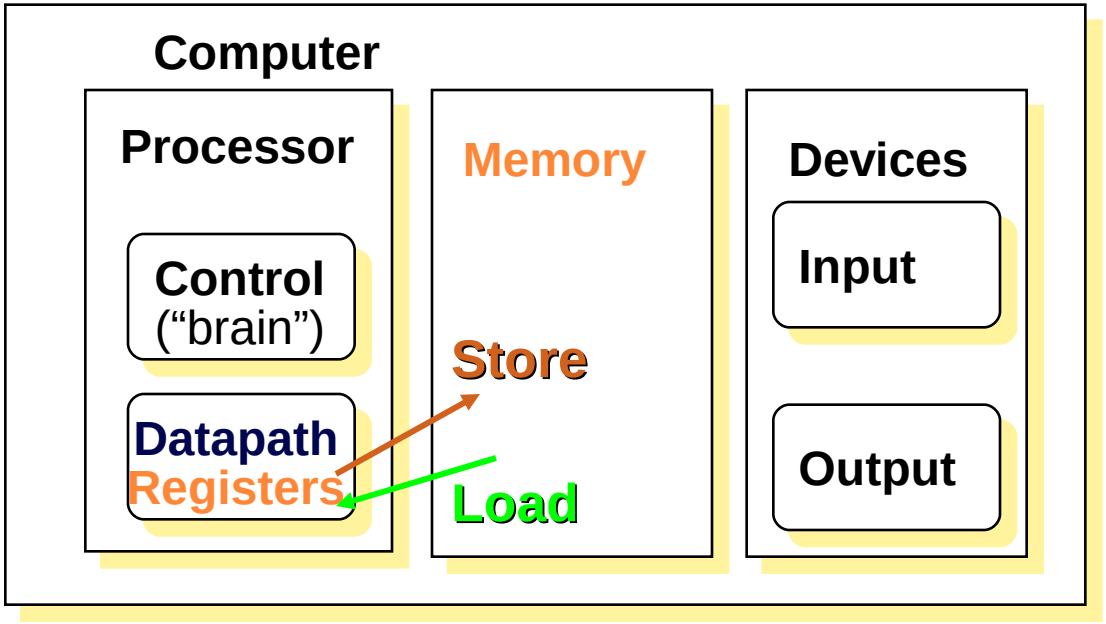
JENIS-JENIS OPERASI (*TIDAK BANYAK BERUBAH SEJAK 1960*)

Data Transfers	**memory-to-memory move** **register-to-register move** **memory-to-register move**
Arithmetic & Logic	**integer (binary + decimal) or FP** **Add, Subtract, Multiply, Divide** **shift left/right, rotate left/right** **not, and, or, set, clear**
Program Sequencing & Control	**unconditional, conditional Branch** **call, return** **trap, return**
Input/Output Transfers	**register-to-i/o device move**
Synchronization **String** **Graphics (MMX)**	**test & set (atomic r-m-w)** **search, translate** **parallel subword ops (4 16bit add)**

REGISTER: MEMORI KHUSUS



Registers are in the datapath of the processor; if operands are in memory, we must transfer them to the processor to operate on them, And then transfer back to memory when done



REGISTER TRANSFER NOTATION

- Notasi yang menggambarkan proses pertukaran data:
  - arah: dari sumber ke tujuan
  - operasi: '+', '-', …
- Sumber/Tujuan Data:
  - Register → R0, R5
  - Memori → LOC, PLACE, A, VAR2
  - I/O Device → DATAIN, OUTSTATUS
- Contoh:
  - Pertukaran data:

    R1 ← [LOC] ; isi lokasi memori 'Loc' di-
    ; *copy*-kan ke register R1

  - Operasi:

    R3 ← [R1] + [R2] ; isi register R1 dijumlahkan
    ; dengan isi register R2,
    ; hasilnya disimpan di
    ; register R3

ASSEMBLY LANGUAGE NOTATION

- Notasi yang menggambarkan program dalam bahasa mesin (agar lebih mudah dipahami)

- Jenis-jenis Operasi:
  - Transfer Data: Move, Load, Store
  - Aritmatika & Logika: Add, Sub, And, Or, …
  - Kendali: Beq, Bne, Jmp, Call, Ret, …
  - Transfer I/O: In, Out, …

- Contoh:

```
Move        LOC,R1      ; R1 ← [LOC]
Add         R1,R2,R3    ; R3 ← [R1] + [R2]
```

KELAS-KELAS INSTRUKSI BAHASA MESIN

- 3 address

  Add A,B,C ; C ← [A] + [B]

  Operation Source1,Source2,Destination

  atau

  Operation Destination,Source1,Source2

- **2 address**

  Add A,B ; B ← [B] + [A]

  Operation Source,Destination

- **1 address**

  Add A ; acc ← [acc] + [A]

- **0 address**

  Add ; tos ← [tos] + [next]

- **Load/Store + General Purpose Register:**
  - **2 address**

    Load A,R1 ; R1 ← [B]

    Load B,R2 ; R2 ← [A]

    Add R1,R2 ; R2 ← [R2] + [R1]

    Store R2,C ; C ← [R2]

PERBANDINGAN "PROGRAM"

- Perintah HLL: C = A + B
  - Isi lokasi memori A & B tidak boleh berubah

- 3-address:

  Add A,B,C ; C ← [A] + [B]

0-addres	1-address	2-address	3-address (load-store)
Push A	Load A	Move B,C	Load A,R1
Push B	Add B	Add A,C	Load B,R2
Add	Store C		Add R1,R2,R3
Pop C			Store R3,C