BAHASA RAKITAN

Universitas Gunadarma

Kuliah I : Pengenalan

I. Pengenalan (cont'd)

I.1. Bahasa

- Komputer "bicara/berkomunikasi" dgn suatu bahasa
- Bahasa-2 pemrograman menyediakan tools u/ mengekspresikan pemrosesan data secara simbolik
- Setiap bahasa memiliki sintaks dan grammar yang dirumuskan dgn baik

I.2. Bahasa-2 Pemrograman

- Terdapat banyak bahasa pemrograman → tergolong menjadi 2:
  - **Bahasa Tingkat Tinggi: (C, C++, Pascal, Basic)**
    - Machine-independent dan instruksinya lebih ekspresif (spt bhs manusia)
  - **Bahasa Tingkat-rendah: (bahasa rakitan)**
    - Machine-specific; dan instruksinya lebih halus-kecil (finer-grained) yang terkait erat dgn bahasa mesin dari prosessor target.

I. Pengenalan (cont'd)

I.3. Bahasa-2 Rakitan

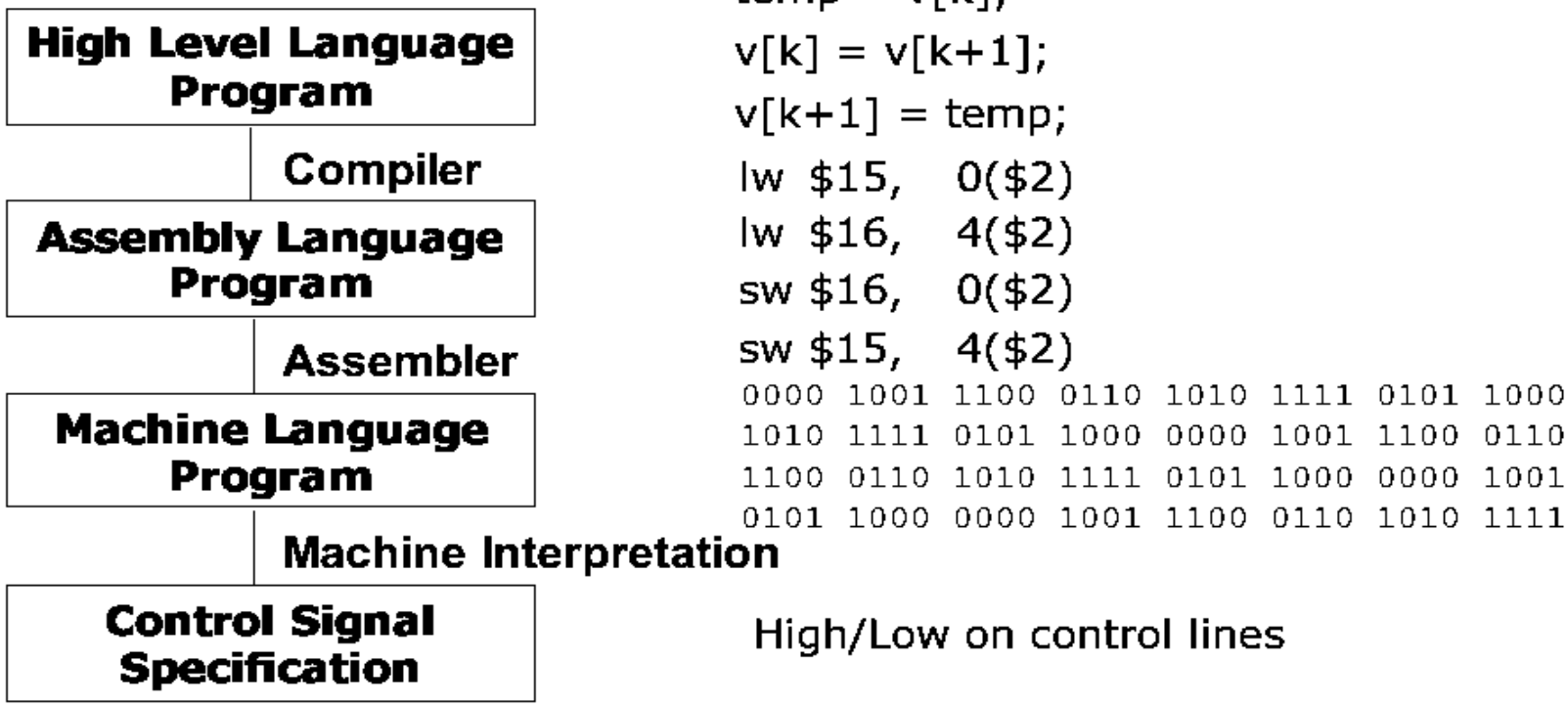
- Bahasa-2 rakitan → representasi teks dari bahasa mesin
- Satu statement merepresentasikan satu instruksi mesin
- Lapisan abstraksi antara program-2 tingkat tinggi dan kode mesin

I.4. Bahasa Mesin

- Merupakan bahasa ibu/alamiah dari komputer
- Kata-2 (WORDS) → **instruksi-2**
- Vocabulary → **set instruksi**
- Representasi bit dari operasi mesin dieksekusi oleh hardware

I. Pengenalan (cont'd)

- **Hirarki bahasa-2 diatas sbb:**



I. Pengenalan (cont'd)

Catatan:

- Bahasa rakitan membuka rahasia HW dan SW komputer
  - Mempelajari/menganalisa :
    - tentang cara HW komputer dan Sistem operasi bekerja sama;
    - bagaimana program aplikasi berkomunikasi dgn sistem operasi
- Tidak terdapat bahasa rakitan tunggal.
  - Setiap komputer atau family komputer menggunakan set instruksi mesin yang berbeda dan bahasa rakitan yang berbeda
    - Bahasa rakitan IBM-PC mengandung hanya set instruksi intel 8086/8088, dgn enhancement u/ 80186/286/386 (instruksi set 8088 dapat digunakan tanpa modifikasi yang signifikan)
- **Assembler:** suatu program yang mengkonversi program source-code ke bahasa mesin:
  - IBM-PC → Microsoft Macro Assembler (MASM), Borland's Turbo Assembler
  - Focus → MASM (mengenal lebih dari 50 directives)

  Jadi bahasa rakitan IBM-PC merujuk set instruksi 8086/8088 dan set komplit dari directives MASM

I. Pengenalan (cont'd)

Mengetahui Bahasa Rakitan perlukah?

- Dipelajari dgn berbagai alasan:
  - Mengetahui lebih dalam tentang arsitektur komputer dan sistem operasi
  - Mengetahui lebih lanjut tentang komputer dan bagaimana bahasa komputer membangkitkan kode mesin

    Jelas: karena bhs rakitan mempunyai hubungan yang dekat dgn bhs mesin
  - Mempelajari ***utilitasnya***.
    - Tipe pemrogramman tertentu sulit atau tidak mungkin dilakukan dgn bhs tingkat tinggi. Contoh:
      - Komunikasi langsung dgn SO komputer
      - Program color high-speed graphics dgn memori rendah
      - Program interfacing
      - Program telekomunikasi

I. Pengenalan (cont'd)

- Sebagai Solusi akibat batasan-2 pada bhs tingkat tinggi.
  - Contoh:
    - Bhs tingkat tinggi: tidak diizinkan pengerjaan (assigning) sebuah nilai karakter ke variabel integer. (buat programmer yang expert no problem, namun programnya tidak kompetibel)
  - Bhs rakitan batasan atau aturannya sangat sedikit
- Sebagai alat belajar (learning tool) → terutama menyakut kerja OS

Aplikasi bhs. Rakitan

- Program subroutine: aplikasi spesifik (short program)
  - Program subroutine (dpt dipanggil oleh bhs tingkat tinggi)
    - Kombinasi ini → diperoleh kekuatan bhs tingkat tinggi
  - Contoh: Program COBOL; andikan compilernya tidak mendukung akan kebutuhan mis: u/pengecekan harddisk, pembuatan subdirektori, proteski file, dll.
    - Tugas ini dapat diatasi dgn membuat subroutine bhs rakitan

I. Pengenalan (cont'd)

I.5. Instruksi-2 bahasa rakitan

- Tipe instruksi dasar memiliki 3 komponen (4 komponen dalam mesin MIPS):
  - Nama operator
  - Tempat menyimpan (store) hasil
  - Operand 1
  - Operand 2 (dalam MIPS)

  Contoh: `mov al,5 (intel)`

  `add a,b,c (MIPS)`
- Format-format yang fix dan sederhana → membuat implementasi HW lebih sederhana ("simplicity favors regularity")
- Pada kebanyakan arsitektur, tidak ada batasan akan elemen-2 yang muncul lebih dari satu

I. Pengenalan (cont'd)

I.5.1. Mnemonic

- Merupakan kode alphabet pendek yang membantu memori dalam mengingat suatu instruksi CPU.
- Dapat berupa: ***instruksi*** atau ***directive***
- Contoh: instruksi: `mov` (move)
  directive: `DB` (define byte)

I.5.2. Operand

- Sebuah instruksi bisa berisi nol, satu atau dua operand
- Sebuah Operand bisa berupa: ***register***, ***variabel memori***, *atau* ***immediate value***
- Contoh:

`10`	(immediate value)
`count`	(variabel memori)
`AX`	(register)

I.5.3 Komentar

- Tanda # dalam MIPS
- Tanda ; dalam intel

I. Pengenalan (cont'd)

I.6. Unsur-2 Dasar Bhs Rakitan

- **Set karakter dasar (pada MASM)**

  Letter: A-Z, a-z

  Digits: 0-9

  Karakters khusus:

?	@	_	$	:
. (period)	[ ]	( )	<>	,(comma)
"(double quotes)	&	%	!	` (appostrope)
|	\	=	{ }	# +

- **Pembangun dasar statemen bhs rakitan :**
  - **Konstanta** (nilai yg tidak berubah saat runtime) → bilangan atau karakter
  - **Variabel** (lokasi penyimpanan yg dapat berubah saat runtime)
  - **Name** → mengidentifikasikan label, variabel, simbol atau reserved word
  - **Mnemonic**
  - **Operands**
  - **Kommentar**

I. Pengenalan (cont'd)

- **Format Statement (pada MASM)**

  `[name] [mnemonic] [operands] [;comments]`

Statement dibagi menjadi 2 kelas:

- ***Instruksi :*** executable statement
- ***Directive :*** statement yang menyediakn informasi untuk membantu assembler dalam menghasil *executable code.*

Statement bersifat *free-form* →

- Ditulis dalam suatu kolom dgn sejumlah spaces antar setiap operands
- Blank lines diizinkan
- Dapat ditulis dalam suatu baris tunggal (max. 128 kolom)

- **Catatan:**
  - Directives: statement yang mempengaruhi baik listing program maupun cara kode mesin dibangkitkan
    - Contoh: DB directive → memerintahkan MASM u/ meng-create storage untuk variabel bernama **count** dan menginisialisasikan dgn 50

      ```
      count db 50
      ```
  - Instruksi: dieksekusi oleh Mikroprosessor saat runtime.
    - Tipe umum: ***program control, data transfer, arithmetic, logical, dan I/O***

I. Pengenalan (cont'd)

I.6 Operator Aritmatik

- Misalkan dalam C, operasi penjumlahan:

```
a=b+c;
```

- Operasi penjumlahan dalam MIPS dan intel:

```
add a,b,c          add b,c
                   mov a,b
```

- Operasi pengurangan (a = b – c) dalam MIPS dan intel:

```
sub a,b,c          sub b,c
                   move a,b
```

I.7 Operasi-operasi kompleks

- Misalkan: `a=b+c+d-e;`
- Menjadi (dlm MIPS)

```
add t0,b,c    #t0=b+c
add t1,t0,d   #t1=t0+d
sub a,t1,c    #a=t1-e
```

Catatan
Kompiler-kompiler biasanya menggunakan variabel-2 temporal Ketika membangkitan kode

I. Pengenalan (cont'd)

I.8. Representasi data

- Bits: 0 dan 1
- Bit string (sederetan dari bits (sequence of bits))
  - 8 bits → 1 Byte
  - 16 bits → half-word
  - 32 bits → word
  - 64 bits → double-word
- Karakter-2 – satu byte, biasanya menggunakan ASCII
- Bilangan integer – disimpan 2's complement
- Floating point – menggunakan suatu *mantissa dan ekponensial*

  $(m \times 2^e)$

I. Pengenalan (cont'd)

I.9. Penyimpan Data (data Storage)

- Pada Bahasa tingkat tinggi, data disimpan dalam variabel
- Dalam prakteknya, data disimpan pada banyak tempat/media yang berbeda:
  - Disk
  - Random Access Memory (RAM)
  - Cache (RAM atai disk)
  - Registers

I. Pengenalan (cont'd)

I.10. Organisasi Register

- Organisasi register merupakan satu aspek yang menentukan tentang "Arsitektur Prosessor" tertentu
- Tiga mekanisme dasar untuk Operator/operand
  - **Akkumulator** : arsitektur yang menggunakan suatu register tunggal u/ satu dari sources dan destination (contoh 8088)
  - **Stack** : operand dipushed dan di-pop (contoh java machine)
  - **General purpose** : sejumlah terbatas register digunakan u/ menyimpan data u/ setiap maksud/tujuan (purpose) (kebanyakan sistem saat ini)
- Register: blok memori kecil kecepatan tinggi (small high-speed) yang digunakan u/ menyimpan data
- Focus → general purpose register

I. Pengenalan (cont'd)

Contoh Akkumulator

- Misalkan: `A=B+C;`
- Dalam arsitektur berbasis-akkumulator, menjadi:

```
load   addressB
add    addressC
store  addressA
```

Contoh Stack

- Misalkan: `a=b+c;`
- Dalam Javacode

```
iload  a      #load b ke stack
iload  c      #load c ke stack
iadd          #add dan puts hasil ke stack
istore a      #store ke a
```

I. Pengenalan (cont'd)

Register General Purpose (GP)

- Pada arsitektur yang menggunakn register GP, data dapat diakses dalam cara-cara yang berbeda
  - **Load/Store** (**L/S**) – data disimpan ke register-register, dioperasikan di dalamnya, dan simpan balik ke memori (contoh. Seluruh set-set instruksi RISC)
    - **Hardware u/operand-2 → sederhana**
    - **Semakin kecil semakin cepat, karena "clock cycle" dapat menjadi dan dipertahankan lebih cepat**

  ⇒ **Penekanan pada Efisiensi**

  Memori-Memori – operand-2 dapat menggunakan alamat memori (memori addresses) sebagai keduanya sources dan destination (contoh INTEL)