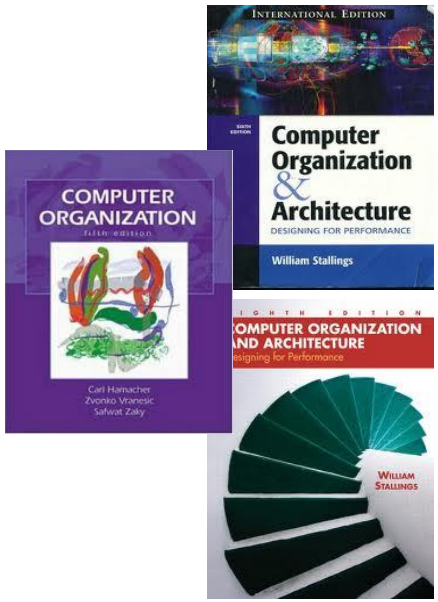
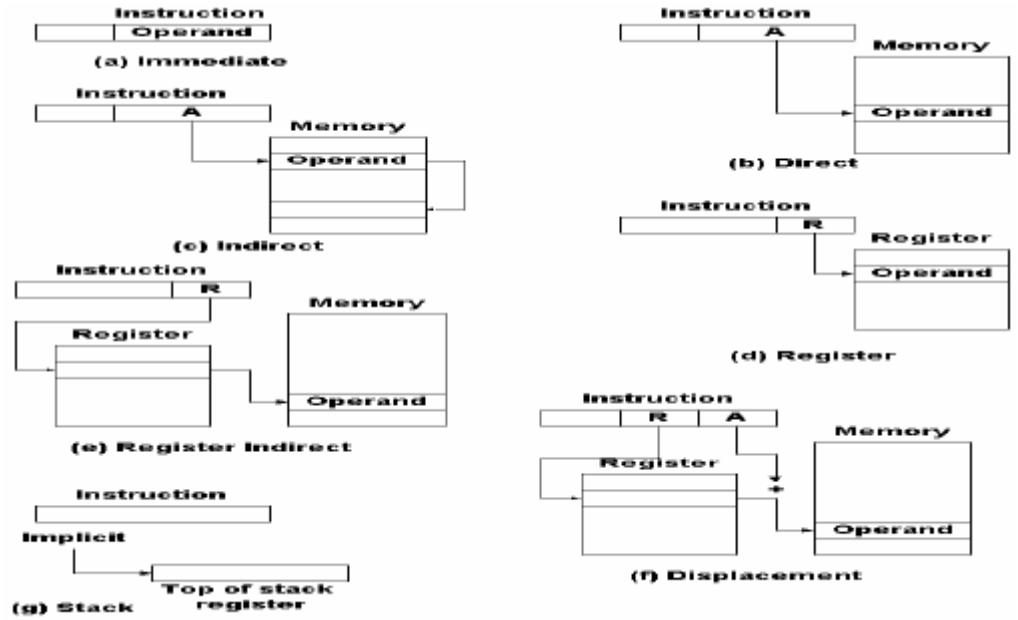
Organisasi & Arsitektur Komputer

Set Instruksi





Eko Budi Setiawan, S.Kom., M.T.

Arsitektur Komputer

Latar Belakang ?

Ada beberapa parameter penting dari CPU yang berpengaruh pada kinerja dan produktivitas sistem.

Arsitek Komputer berfokus pada perancangan set instruksi. Adanya kelemahan pada desain set instruksi akan mempengaruhi secara drastis programmer sistem komputer, baik itu hardware ataupun software.

CISC Vs RISC

Perbandingan CISC dan RISC

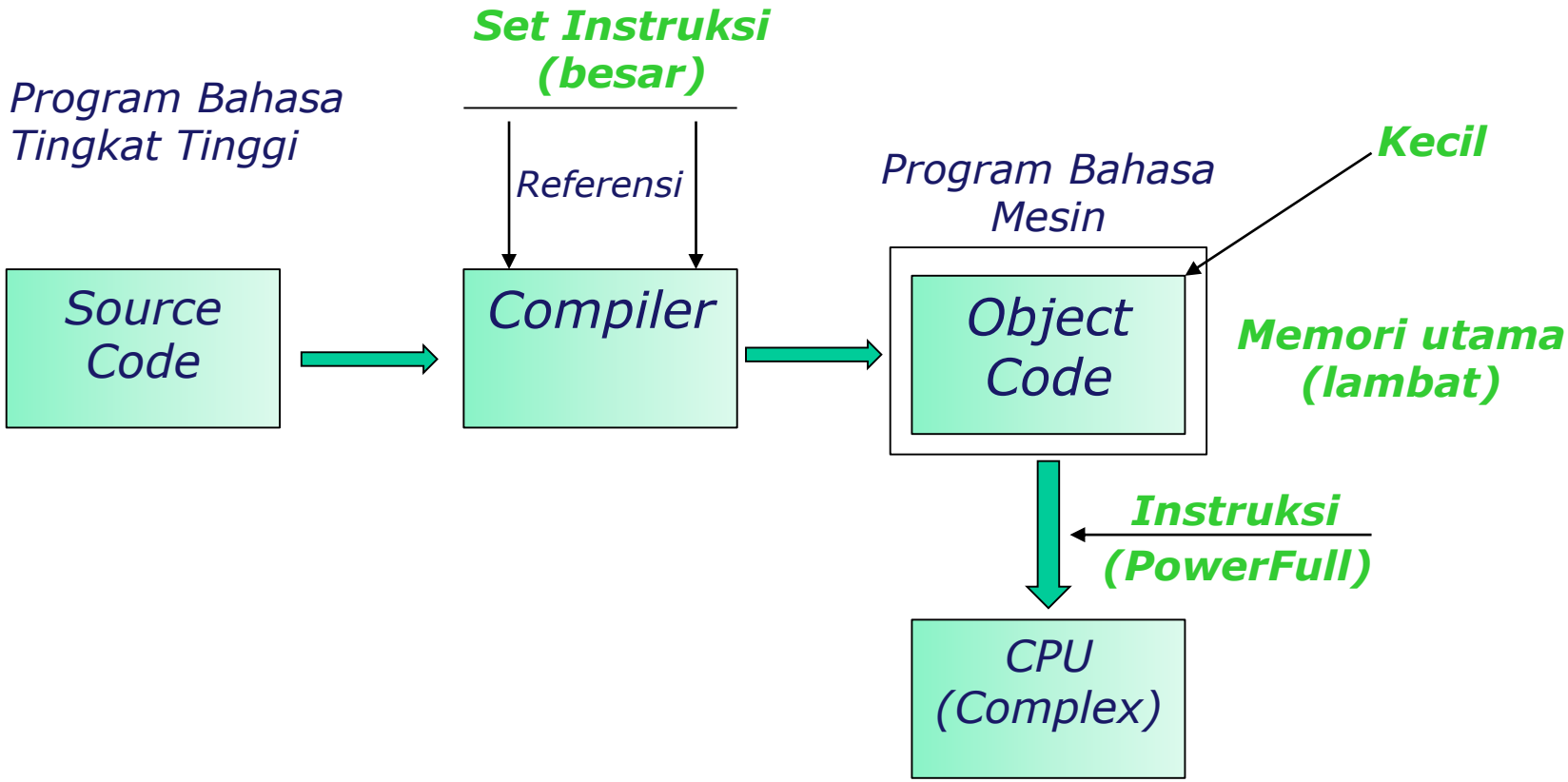
Complex Instruction Set Computing (CISC)

Semua sistem yang lama baik itu komputer mainframe atau personal komputer relatif mempunyai sistem CISC. Set Instruksi dibuat kompleks.

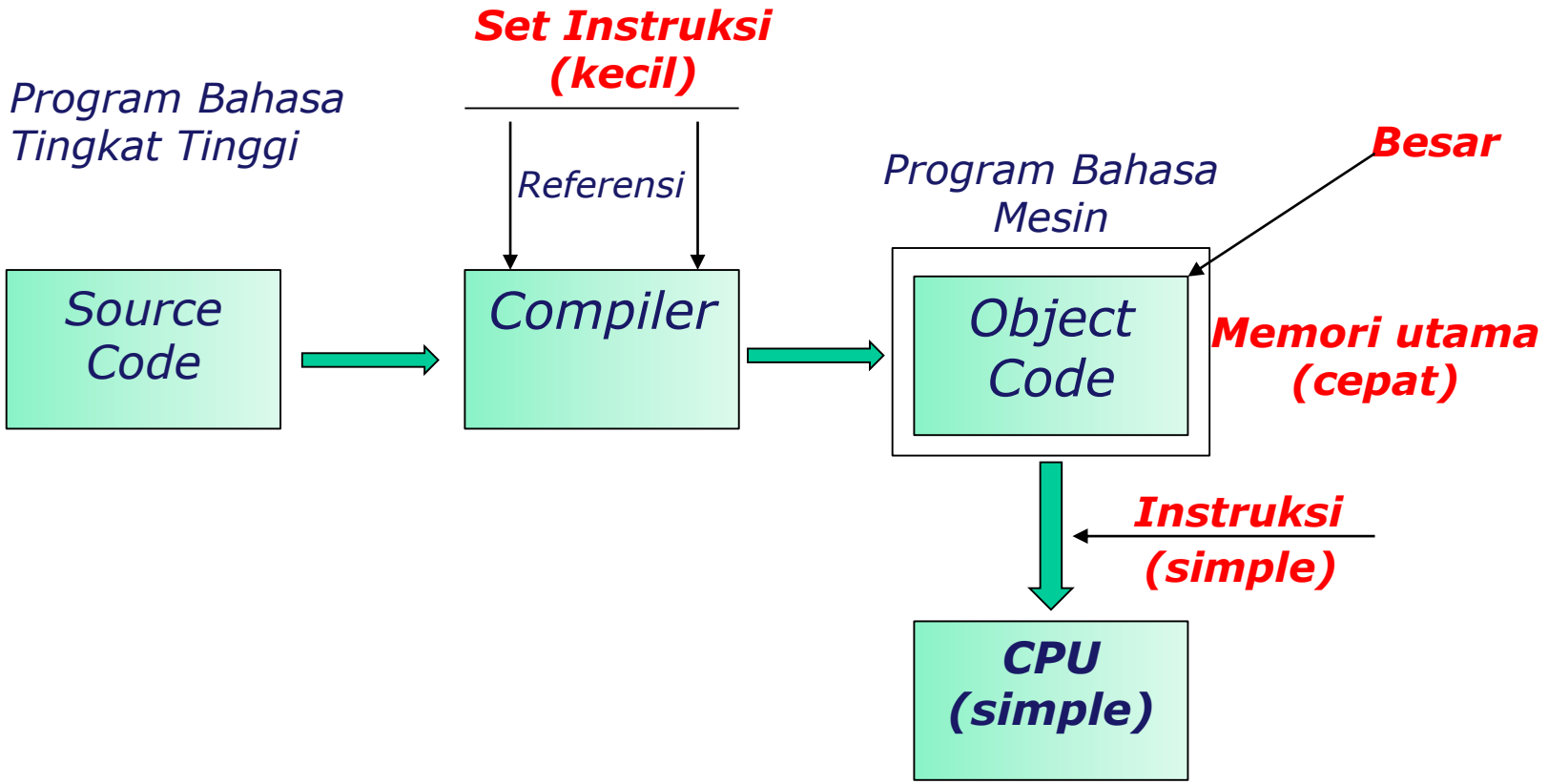
Reduce Instruction Set Computing (RISC)

Sistem RISC lebih populer karena tingkat kinerjanya, dibandingkan dengan CISC. Namun karena biaya yang tinggi, sistem RISC hanya digunakan ketika diperlukan kecepatan khusus. Set Instruksi dibuat simple.

Complex Instruction Set Computing (CISC)



Reduce Instruction Set Computing (RISC)



Kelemahan CISC

Kelemahan CISC

Kompleksitas CPU

Desain kode instruksi menjadi kompleks karena mempunyai set instruksi yang besar

Ukuran Sistem dan Biaya

Mempunyai banyak sirkuit hardware menyebabkan CPU menjadi kompleks. Hal ini meningkatkan kebutuhan daya listrik

Kecepatan Clock

Karena sirkuit yang besar maka waktu delay juga menjadi lebih besar

Keandalan

Dengan hardware yang besar maka cenderung mudah terjadi kegagalan

Konsep RISC

Keep is short and simple

Arsitektur RISC mempunyai fitur sebagai berikut :

Instruksinya sederhana –
Set instruksi kecil –
Panjang instruksinya sama untuk semua instruksi –
Register untuk penyimpanan operand jumlahnya besar –
Arsitektur Load/Store tersedia di register, bukan di memori –
Eksekusi instruksi yang lebih cepat –

Desain Set Instruksi

Pekerjaan yang paling penting dan kompleks dalam mendesain komputer adalah membuat set instruksi

Kenyamanan Pemrograman

Programmer lebih suka mempunyai sebanyak mungkin instruksi supaya operasi yang tepat dapat dikerjakan oleh rangkaian instruksi.Tetapi terlalu banyak instruksi menjadi kompleks dan memerlukan waktu yang besar

Pengalamatan yang fleksibel

Programmer akan suka apabila memungkinkan semua mode pengalamatan operand ada didalam arsitektur. Hal ini memberikan fleksibilitas yang banyak kepada pemrogram.

Jumlah General Purpose Register (GPR)

Jika CPU mempunyai register yang banyak, programmer memperoleh pemrosesan dan transfer data yang cepat. Tetapi biaya perangkat keras CPU meningkat dengan banyaknya GPR

Desain Set Instruksi

Pekerjaan yang paling penting dan kompleks dalam mendesain komputer adalah membuat set instruksi

Target Segmen Pasar

Sasaran bidang aplikasi untuk komputer memerlukan operasi-operasi khusus untuk pemrosesan data yang efisien. Komputer scientific harus mempunyai aritmatika floating-point yang tingkat presisinya baik. Sedangkan komputer bisnis harus mendukung aritmetika desimal, dan komputer hiburan harus mempunyai operasi multimedia

Kinerja Sistem

Jika sebuah program mempunyai instruksi sedikit, maka kinerja sistem meningkat karena waktu yang digunakan oleh CPU dalam pengambilan instruksi akan berkurang

Bahasa Rakitan (Assembly)

Bahasa rakitan terdiri atas sebuah rangkaian assembly statement dimana statement (pernyataan) ditulis satu per setiap baris. Setiap baris dalam sebuah program rakitan dibagi menjadi empat field

Label (optional)	OpCode (disyaratkan)	Operand (disyaratkan untuk beberapa instruksi)	Komentar (optional)

Label digunakan untuk menyediakan nama-nama simbolik untuk alamat-alamat memori

Opcode Berisi sebuah singkatan simbol unguk operasi yang akan dikerjakan

Operand Digunakan untuk menetapkan konstanta, label, data, register atau sebuah alamat

Komentar Menyediakan tempat bagi dokumentasi untuk menjelaskan apa yang dikerjakan

Bahasa Rakitan (Assembly)

Mnemonic merupakan sebuah singkatan yang merepresentasikan instruksi bahasa mesin

Mnemonic	Operand	Arti Instruksi
LD	x	Baca operand dari memori (lokasi x) kedalam Accumulator (AC)
ST	x	Simpan isi dari AC kedalam memori (lokasi x)
ADD	x	Tambahkan AC dengan X
SUB	x	Kurangi AC dengan x

Pemilihan Set Instruksi

Pemilihan set instruksi untuk suatu komputer bergantung pada cara CPU yang disusun. Secara tradisional ada tiga organisasi CPU dengan konsep instruksi

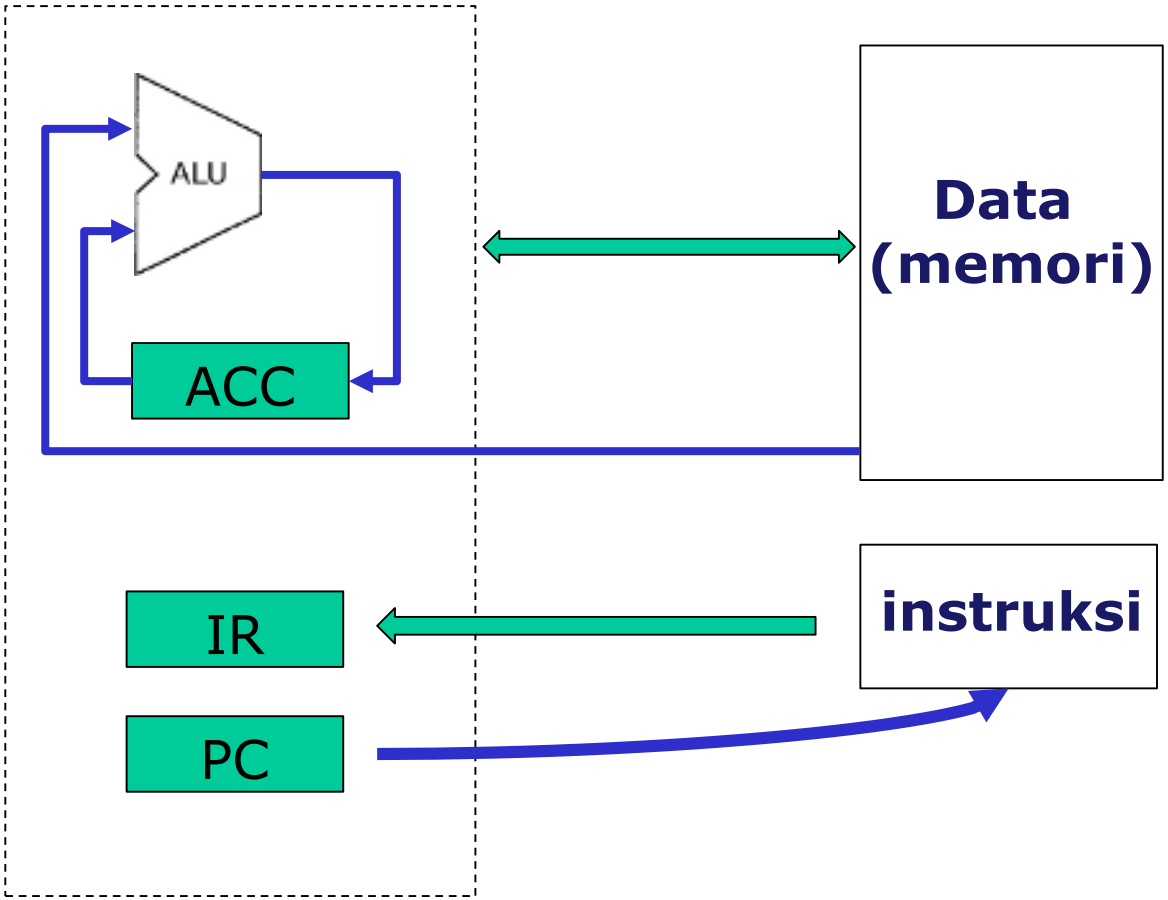
CPU Berbasis Akumulator

CPU Berbasis Register

CPU Berbasis Stack

CPU Berbasis Akumulator

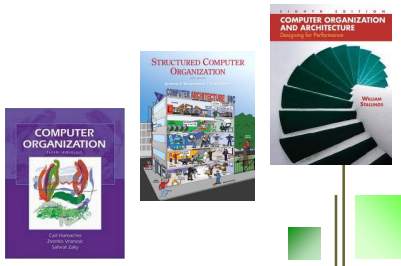
Pada mulanya komputer adalah berbasis akumulator. Akumulator berisi satu operand pada instruksi, demikian juga hasilnya disimpan pada akumulator.



Contoh Program CPU Berbasis Akumulator

Tuliskan sebuah program bahasa rakitan dalam arsitektur CPU akumulator untuk menyelesaikan statemen
X = (A+B) – (C+D)

LD C *: Salin C ke dalam akumulator*
ADD D *: Jumlahkan D dengan isi akumulator dan hasilnya disimpan di akumulator (akumulator berisi C+D)*
ST X *: Simpan hasil C+D dalam lokasi X*
LD A *: Salin A ke dalam akumulator*
ADD B *: Jumlahkan B ke dalam akumulator dan simpan hasilnya didalam akumulator (akumulator berisi A+B)*
SUB X *: Perkurangkan isi akumulator dengan X dan simpan hasilnya dalam akumulator*
ST X *: Simpan isi akumulator ke dalam lokasi memori X*



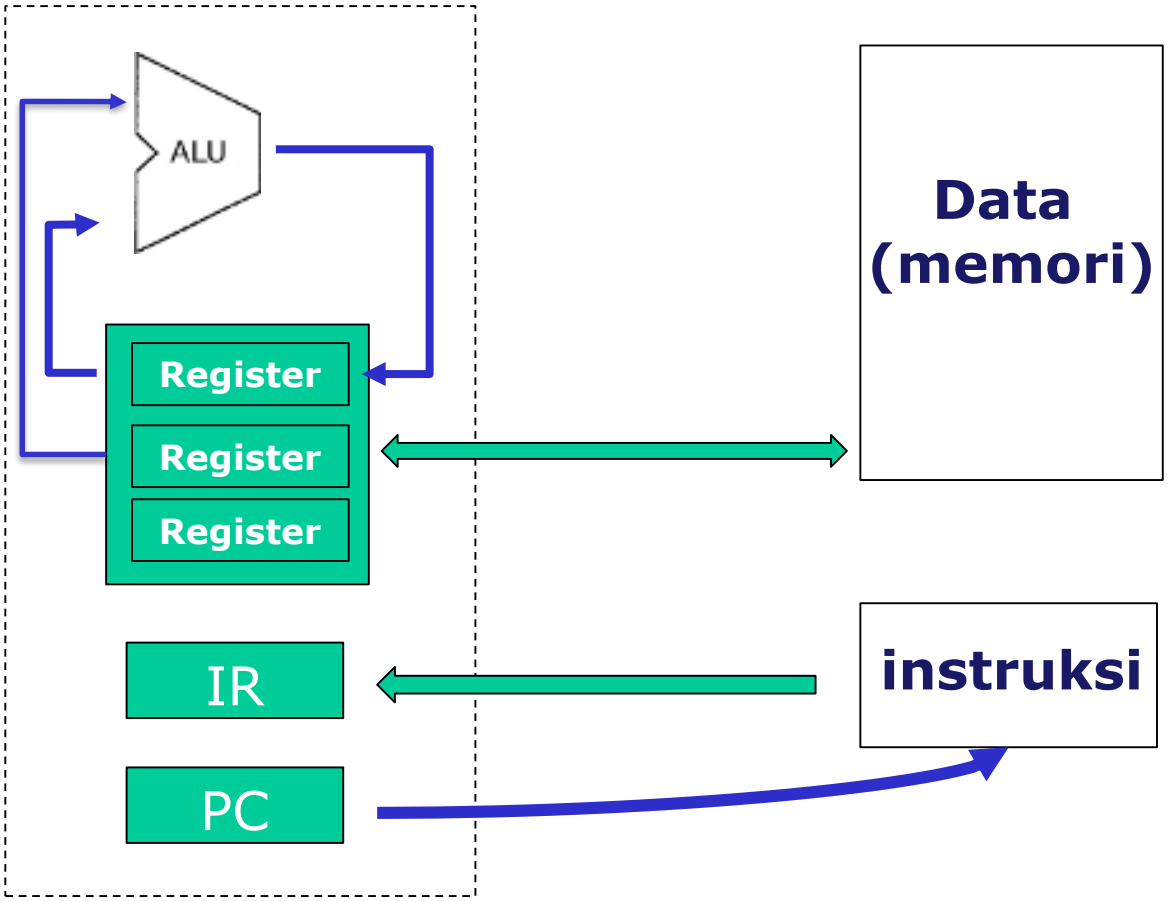
Keuntungan dan Kekurangan CPU Berbasis Akumulator

Keuntungan dan Kekurangan

Kelebihan	Kekurangan
Isi akumulator diperuntukkan bagi satu operand, karena itu tidak memerlukan field alamat (untuk satu operand) dalam instruksi.	*Ukuran program menjadi panjang karena banyak menggunakan instruksi yang ekspresinya kompleks.*
Siklus instruksi memerlukan waktu yang singkat, karena dalam pengambilan instruksi tidak ada siklus pengambilan operand	*Waktu eksekusi program bertambah karena banyaknya jumlah instruksi dalam program*

CPU Berbasis Register

Banyak register sebagai akumulator. Penggunaan register-register tersebut menghasilkan program yang pendek dengan instruksi yang sedikit.

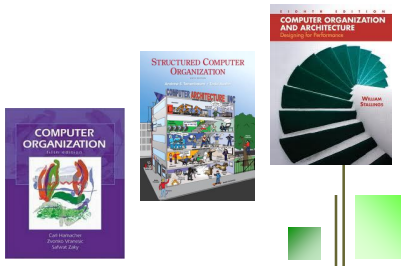


Contoh Program CPU Berbasis Register

Tuliskan sebuah program bahasa rakitan dalam arsitektur CPU register untuk menyelesaikan statemen
X = (A+B) – (C+D)

Penyelesaian menggunakan memori-register

LD R1, A *: Salin A ke dalam register R1*
ADD R1, B *: Jumlahkan isi R1 dengan B dan hasilnya disimpan di R1*
LD R2, C *: Salin C ke dalam R2*
ADD R2, D *: Jumlahkan isi R2 dengan D dan simpan hasilnya dalam R2*
SUB R1, R2 *: Kurangkan isi R1 dengan R2 dan simpan hasilnya dalam R1*
ST R1, X *: Simpan hasil R1 ke dalam lokasi memori X*

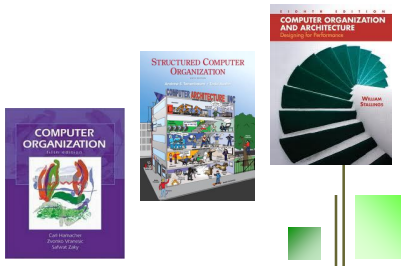


Contoh Program CPU Berbasis Register

Tuliskan sebuah program bahasa rakitan dalam arsitektur CPU register berbasis load store untuk menyelesaikan statemen X = (A+B) – (C+D)

Penyelesaian menggunakan load store register

LOAD R1, A *: Salin A ke dalam register R1*
LOAD R2, B *: Salin B ke dalam register R2*
LOAD R3, C *: Salin C ke dalam register R3*
LOAD R4, D *: Salin D ke dalam register R4*
ADD R5, R3, R4 *: operasi aritmatika R5 ← R3+R4, (R5 = C+D)*
ADD R6, R1, R2 *: operasi aritmatika R6 ← R1+R2, (R6 = A+B)*
SUB R7, R6, R5 *: operasi aritmetika R7 ← R6 - R5,(R7 = (A+B) – (C+D))*
STORE R7, X *: Salin isi R7 ke dalam X*



Keuntungan dan Kekurangan CPU Register – Load Store

Keuntungan dan Kekurangan Mesin Load Store (tiga operand)

Kelebihan

Sederhana - Instruksi menggunakan jumlah siklus yang sama Relatih mudah -

Kekurangan

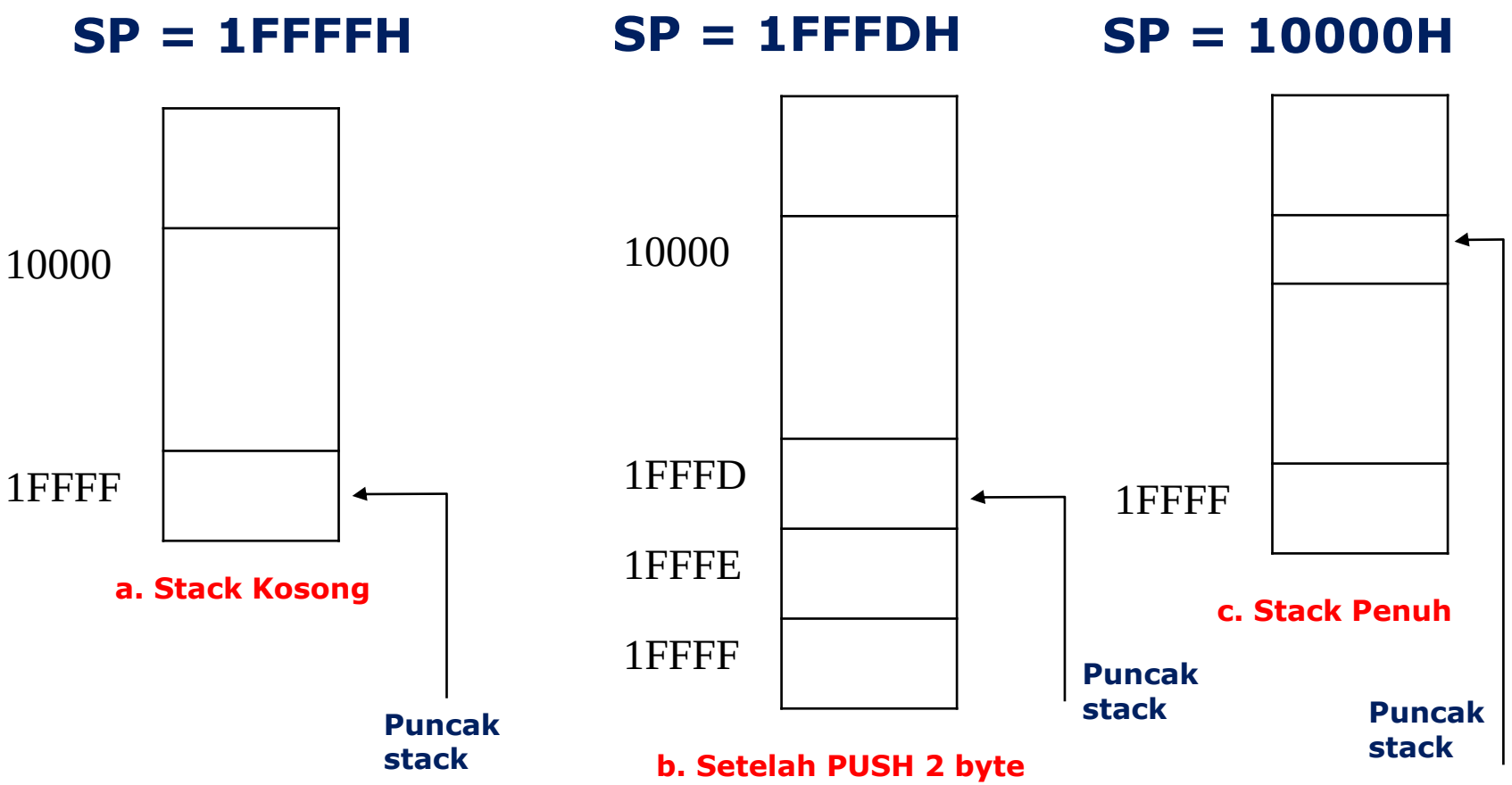
- *Jumlah instruksi yang lebih banyak*
- *Tidak semua instruksi memerlukan tiga operand*
- *Bergantung pada kompiler yang baik*

CPU Berbasis Stack

Stack merupakan daftar yang didorong kebawah dengan mekanisme akses LIFO (Last in First Out). Stack menyimpan operand-operand. Suatu register digunakan untuk menunjuk ke alamat lokasi kosong pada puncak stack. Register ini dikenal dengan Stack Pointer (SP).

POP = suatu item diambil dari Stack
PUSH = suatu item disimpan di stack

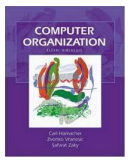
CPU Berbasis Stack



Contoh Program CPU Berbasis Stack

Tuliskan sebuah program bahasa rakitan dalam arsitektur CPU berbasis Stack untuk menyelesaikan statemen
X = (A+B) – (C+D)

Statement	Isi Stack Setelah Eksekusi Instruksi	Lokasi Stack yang diduduki
PUSH A	A	1
PUSH B	A,B	2
ADD	A + B	1
PUSH C	(A+B), C	2
PUSH D	(A+B), C, D	3
ADD	(A+B), (C+D)	2
SUB	(A+B) - (C+D)	1
POP X	Kosong	0



Keuntungan dan Kekurangan CPU Stack

Keuntungan dan Kekurangan

Kelebihan	Kekurangan
Pemrograman mudah/efisiensi kompiler tinggi	*Diperlukan sirkuit hardware tambahan untuk implementasi stack*
Instruksi pendek karena tidak mempunyai alamat	*Ukuran program meningkat*

Panjang Instruksi

Kekurangan Ukuran Instruksi

Apabila Terlalu Pendek

Akan terlalu banyak instruksi dalam program sehingga banyak waktu yang terbuang untuk pengambilan instruksi

Ukuran program bertambah sehingga kebutuhan memori juga semakin bertambah

Apabila Terlalu Panjang

Instruksi menempati ruang memori yang lebih besar, meningkatkan kebutuhan memori sistem

Lebar bus data besar atau pengambilan instruksi lebih memakan waktu

Format Instruksi

Umumnya format instruksi terdiri atas kode operasi dan operand. Suatu instruksi memberikan paling banyak empat informasi pada CPU :

ADD Opcode	Alamat Operand I	Alamat Operand II	Alamat Hasil	Alamat Instruksi Berikutnya

1. Operasi yang akan dikerjakan oleh instruksi
2. Operand (data) yang harus dioperasikan
3. Lokasi (memori atau register) dimana hasil operasi disimpan
4. Lokasi memori dimana instruksi berikutnya harus diambil

Pengalamatan Immediete

OPCode	Operand

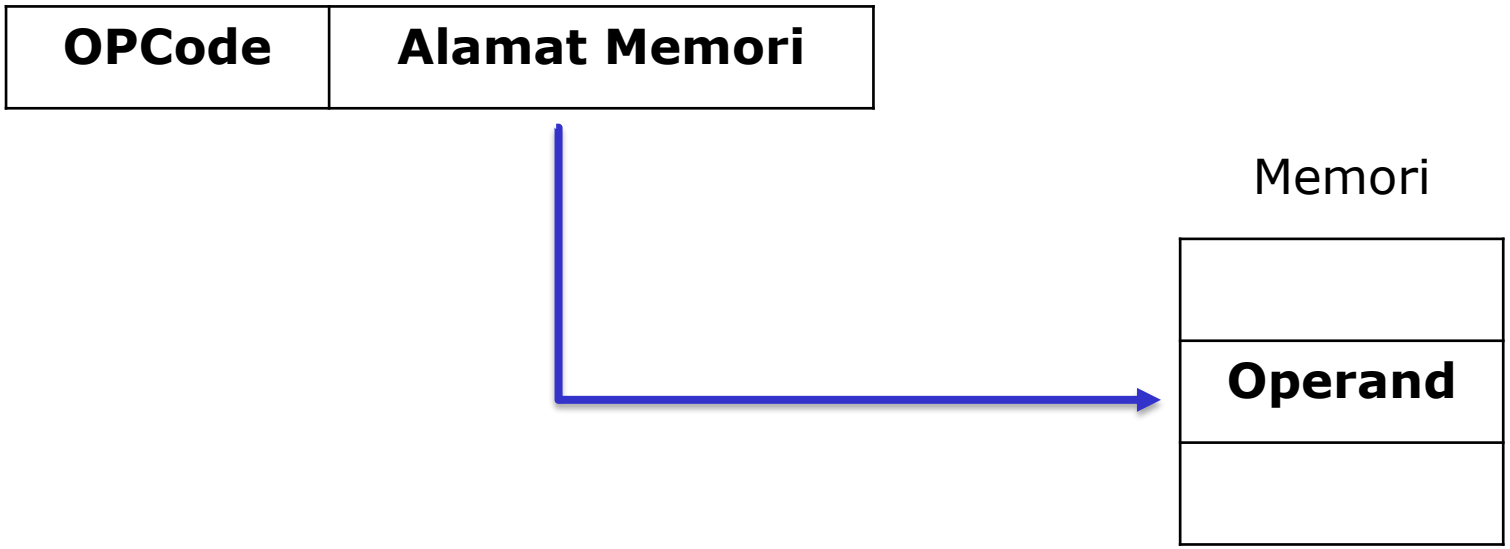
Pengalamatan Immediete

Merupakan mode pengalamatan yang tidak melakukan aktivitas pengambilan operand

Tanda # digunakan untuk menunjukkan bahwa konstanta yang mengikuti tanda tersebut adalah immediate operand

MOVE #26, R1
ADD #26, R1

Pengalamatan Langsung



LOAD R1, X → Salin isi lokasi memori X ke dalam register R1

MOV Y, X → Salin isi lokasi memori X ke dalam lokasi Y

Pengalamatan Tidak Langsung

OPCode	Alamat Tak Langsung (X)

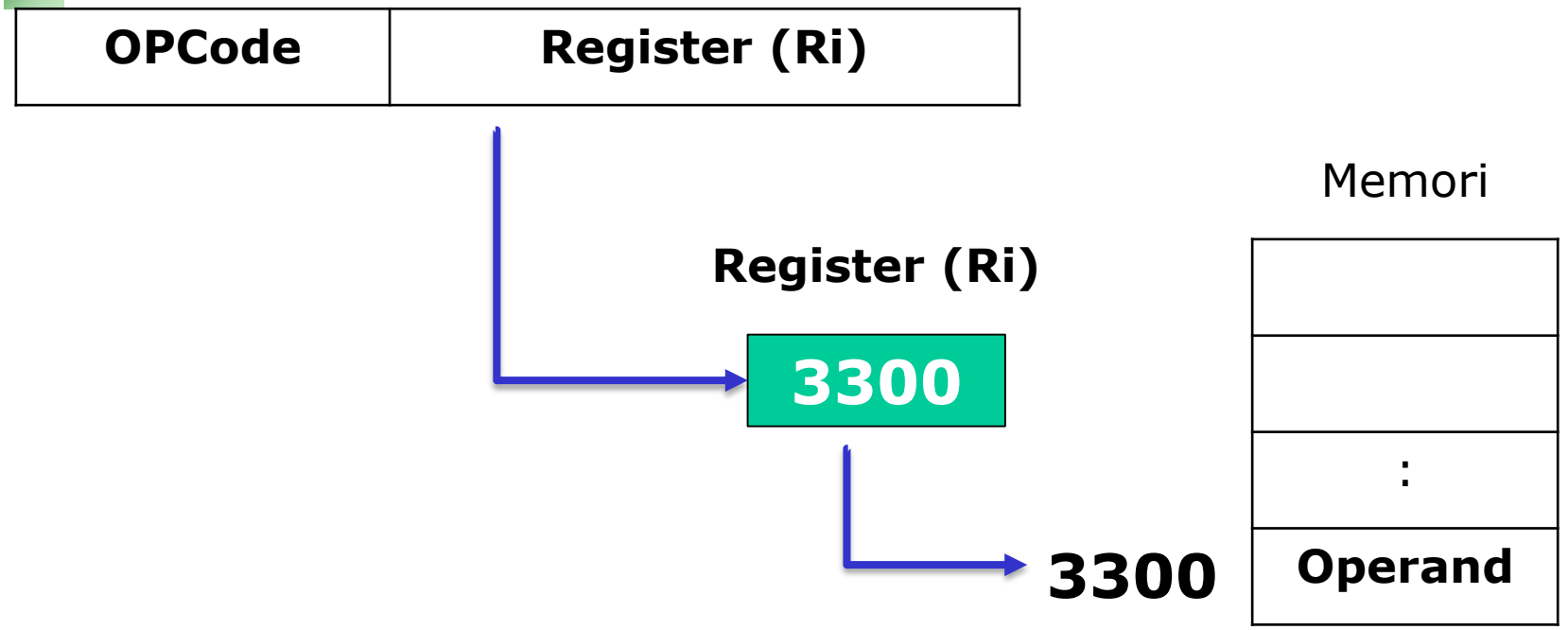
Memori

X	Y
	:
Y	Operand

MOV (X), R1→ Isi dari lokasi yang mempunyai alamat X disalin ke register R

Pengalamatan tak langsung memori
Pengalamatan tak langsung register

Pengalamatan Tidak Langsung Register



Pengalamatan Register

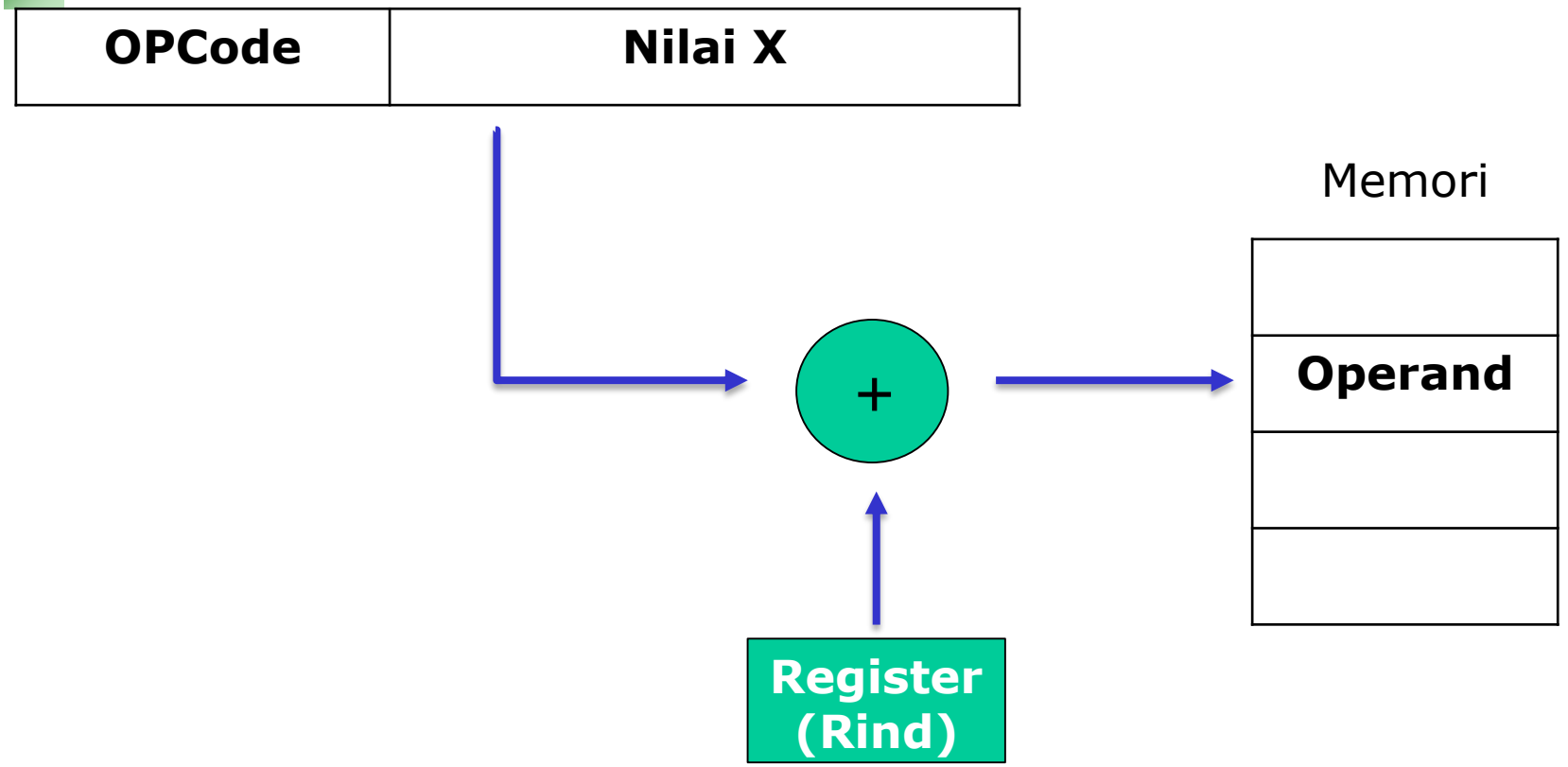
OPCode	Register (R)

Pengalamatan Register

Secara konseptual pengalamatan register mirip dengan pengalamatan langsung, kecuali lokasi memori digantikan dengan register untuk menyimpan operand. Instruksi berisi nomor register yang mempunyai operand.

ADD R1, R2 → Jumlahkan isi register R1 dengan R2 kemudian hasilnya disimpan di R1. Kedua operand menggunakan pengalamatan register.

Pengalamatan Indeks



LOAD X(Rind), Ri→ Instruksi ini menyalin operand alamat hasil penjumlahan nilai X dengan nilai register Rind ke dalam register Ri

Latihan

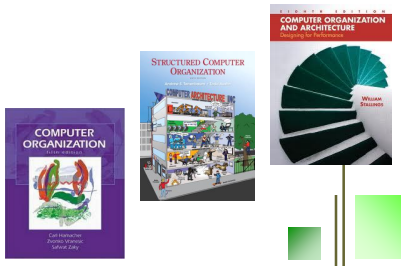
Tuliskan sebuah program bahasa rakitan untuk menyelesaikan statemen

$$X = (A * B) + (C * D)$$

Dengan menggunakan :

a. CPU berbasis akumulator
b. CPU berbasis register
c. CPU berbasis stack

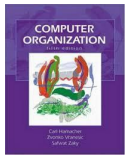
Keterangan :
Perkalian = MUL
Pertambahan = ADD



Jawaban Latihan

X = (A x B) + (C x D)

Berbasis Akumulator	Berbasis register	Berbasis Stack
LD A	LD R1, A	PUSH A
MUL B	MUL R1, B	PUSH B
ST X	LD R2, C	MUL
LD C	MUL R2, D	PUSH C
MUL D	ADD R1, R2	PUSH D
ADD X	ST R1, X	MUL
ST X		ADD
		POP X



Latihan

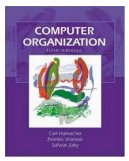
Tuliskan sebuah program bahasa rakitan untuk menyelesaikan statemen

$$Y = (A - B) * (C / D)$$
$$Y = (A - B) / (C + D * E)$$

Dengan menggunakan :

a. **CPU berbasis akumulator**
b. **CPU berbasis register**
c. **CPU berbasis stack**

Keterangan :
Perkalian = MUL
Pertambahan = ADD
Pembagian = DIV



Thank You

To Be Continued..