



PROGRAM STUDI
TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO

MATA KULIAH
ORGANISASI DAN ARSITEKTUR
KOMPUTER

Instruction Set Architecture dan Desain

- ✓ Tipe Instruksi
- ✓ Lokasi dan Operasi Memory
- ✓ Pengantar mode pengalamatan

Tim pengampu

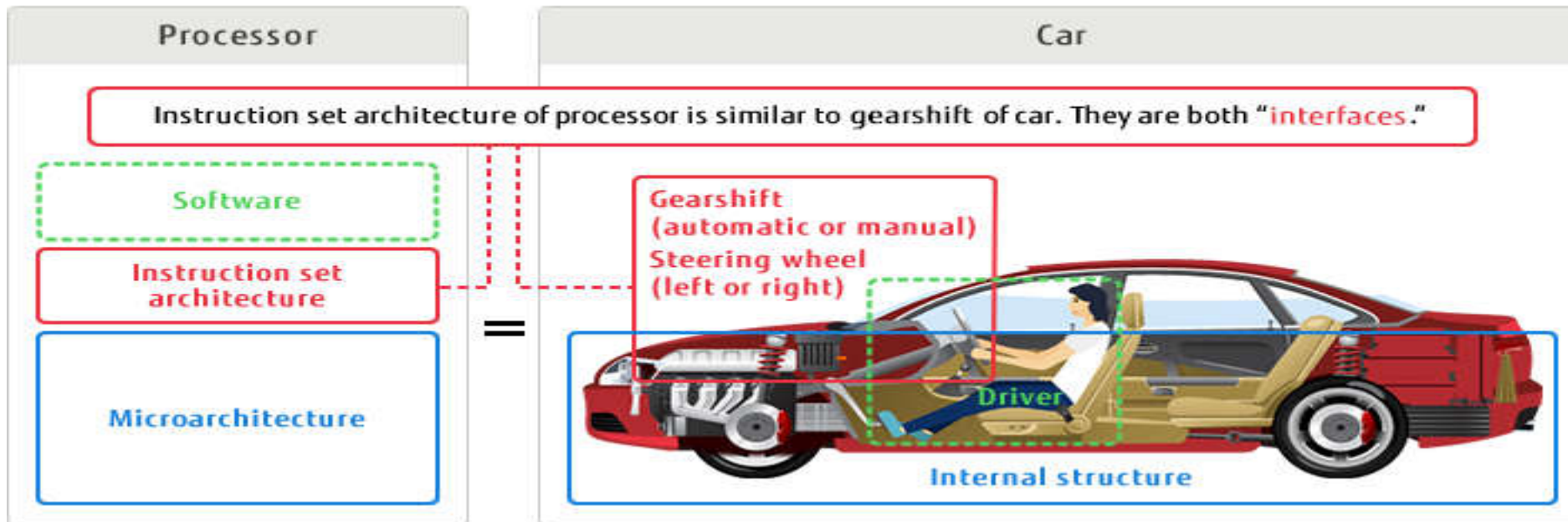
Organisasi dan Arsitektur Komputer

T.A. 2020

Instruction Set Architecture

Bagaimana berkomunikasi dengan komputer?

Instruction Set Architecture and Microarchitecture



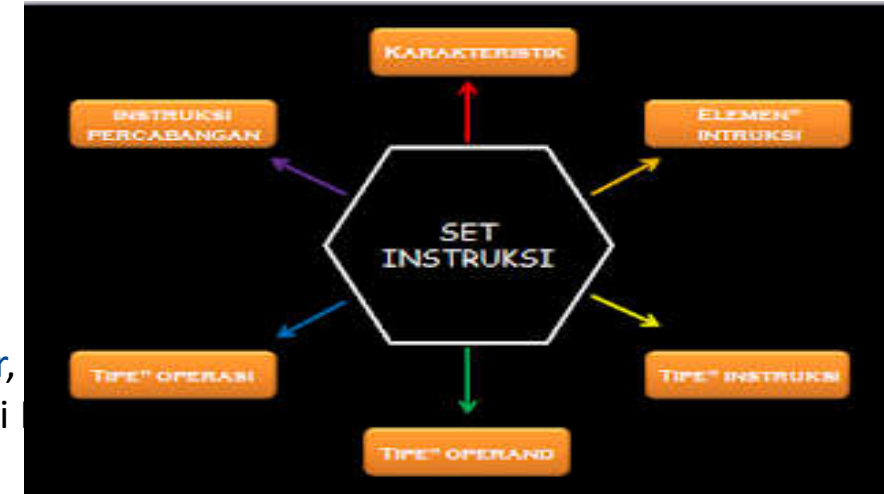
Gambar : <http://journal.jp.fujitsu.com>

Instruction Set Architecture (ISA)

Bagaimana berkomunikasi dengan komputer ?

Instruction Set adalah interface antara hardware dan software prosesor.

- ISA mendefinisikan interface antara hardware dan software prosesor, tetapi tidak menentukan struktur internal prosesor.
- ISA ini mencakup jenis **data** yang didukung, jenis **instruksi** yang dipakai, jenis **register**, **mode pengalamatan**, **arsitektur memori**, penanganan **interupsi**, **eksepsi**, dan operasi eksternalnya (jika ada).
- ISA dapat diemulasikan dalam bentuk **software** oleh sebuah **interpreter**. Lebih lambat dibandingkan dengan menjalankan program secara langsung di atas hardware.



Instruction Set Architecture (ISA)

Bagaimana berkomunikasi dengan komputer ?

Instruction Set adalah perantara antara hardware dan software prosesor.

- Setiap vendor prosesor secara mandiri mendefinisikan struktur prosesor internal mereka sendiri.
- Contoh : Fujitsu dan Oracle mendesain struktur internal dari prosesor sendiri, secara mandiri, tetapi prosesor mereka menggunakan ISA SPARC yang sama.
- Struktur internal prosesor (microarchitecture), sangat mempengaruhi kinerja dan keandalan prosesor, ini adalah teknologi kunci untuk vendor prosesor.

SPARC T5

- 16 S3 cores @ 3.6GHz
- 8MB shared L3 Cache
- 8 DDR3 BL8 Schedulers providing 80 GB/s BW
- 8-way 1-hop glueless scalability
- Integrated 2x8 PCIe Gen 3
- Advanced Power Management with DVFS

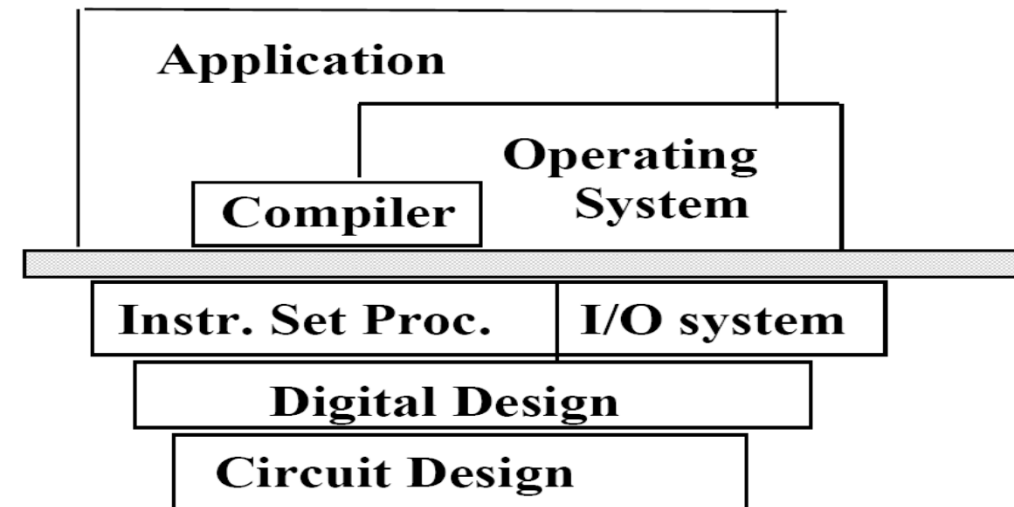


Gambar : Oracle Sparc T5, <http://www.oracle.com>

Instruction Set Architecture

Bahasa mesin dibangun dari pernyataan atau instruksi yang terpisah. Pada **processing architecture**, instruksi yang diberikan dapat menentukan:

- register khusus untuk aritmatika, addressing, atau control functions
- lokasi memori atau offset tertentu
- mode addressing tertentu yang digunakan untuk menafsirkan (interpret) operand

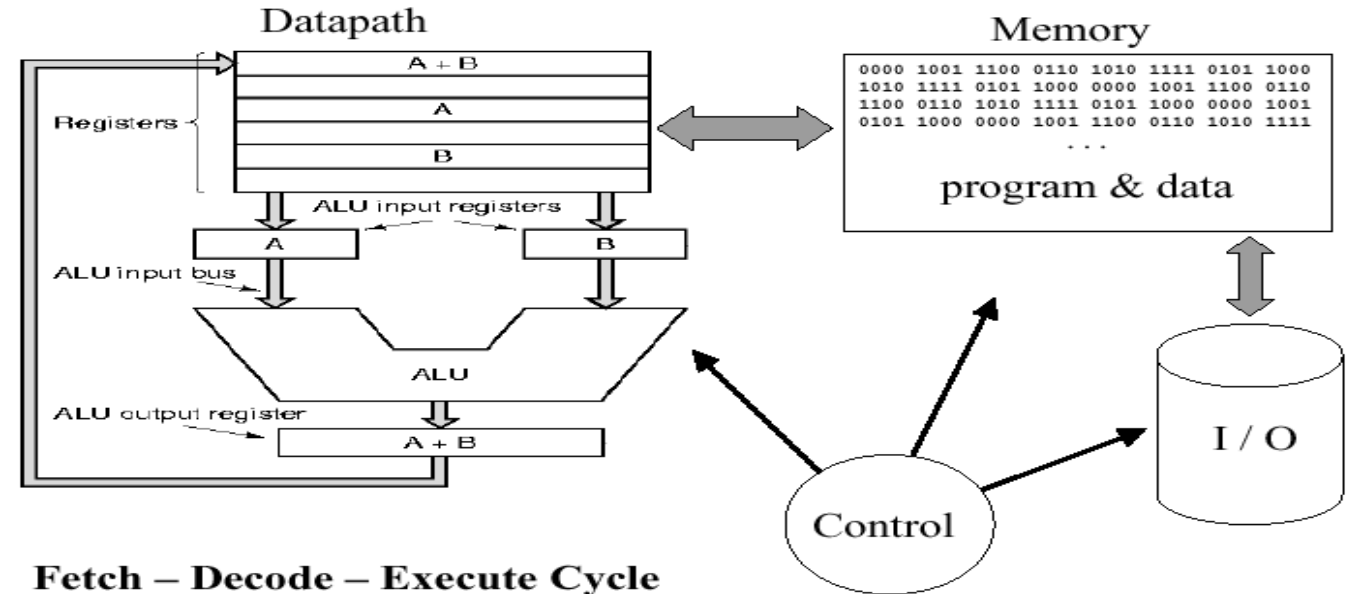


Gambar : allansnavely-L2isa.pdf

Memory Location

Implementasi ISA pada hardware, terdiri dari siklus *fetch-decode-execute* yang diilustrasikan pada Gambar 2.2.

- **fetch**, operand diambil dari memori.
- **decode** menempatkan operand ke dalam format yang dapat dimanipulasi oleh ALU.
- **Execute** melakukan operasi yang dipilih dalam ALU.
- **Control** memfasilitasi perutean data yang teratur, termasuk I / O ke lingkungan eksternal ALU (misalnya, perangkat periferil seperti disk atau keyboard).



Fetch – Decode – Execute Cycle

Gambar : <https://www.cise.ufl.edu/~mssz/CompOrg/CDA-lang.html>

Memory Location

Memory (register) sebagai lokasi penyimpanan *instruksi dan data* dapat dimodelkan menjadi kumpulan kotak2 kosong yang disebut *cell*.

Tiap cell menyimpan muatan listrik yang dapat diasumsikan sebagai *1 bit digit biner* 1 atau 0.

Sejumlah cell dikelola menjadi kelompok-kelompok agar dapat dianggap sebagai satuan terkecil (*atomic entity*) yang mempunyai arti, misalnya *8bit cell*, disebut *byte*.

Proses baca tulis memory biasanya mengambil dan menyimpan sekelompok *atomic entity(byte)*, sejumlah atomic entity disebut sebagai *word (16-64 byte)*. Word adalah satuan terkecil data yang dapat *dialamatkan di memory*

Contoh, memory 256Mb dapat menampung sekitar : $256 \times 2^{20} = 2^{28}$ byte

Type Instruction

Contoh operasi yang umum untuk banyak set instruksi termasuk:

Data handling and memory operations

- Memberikan nilai konstanta ke register
- Copy data dari lokasi memory ke register atau sebaliknya (load and store operations).
- *Read* and *write* data dari hardware devices.

Arithmetic and logic operations

- *Add, subtract, multiply, atau divide* nilai dari 2 register menempatkan hasil di register, setting kondisi pada status register.
- *increment, decrement, membandingkan* 2 nilai register
- *Floating-point instructions* untuk operasi arithmetic pada angka floating-point.

Type Instruction

Contoh operasi yang umum untuk banyak set instruksi termasuk:

Control flow operations

- *Branch* to another location in the program and execute instructions there.
- *Conditionally branch* to another location if a certain condition holds.
- *Indirectly branch* to another location.
- *Call* another block of code, while saving the location of the next instruction as a point to return to.

Coprocessor instructions

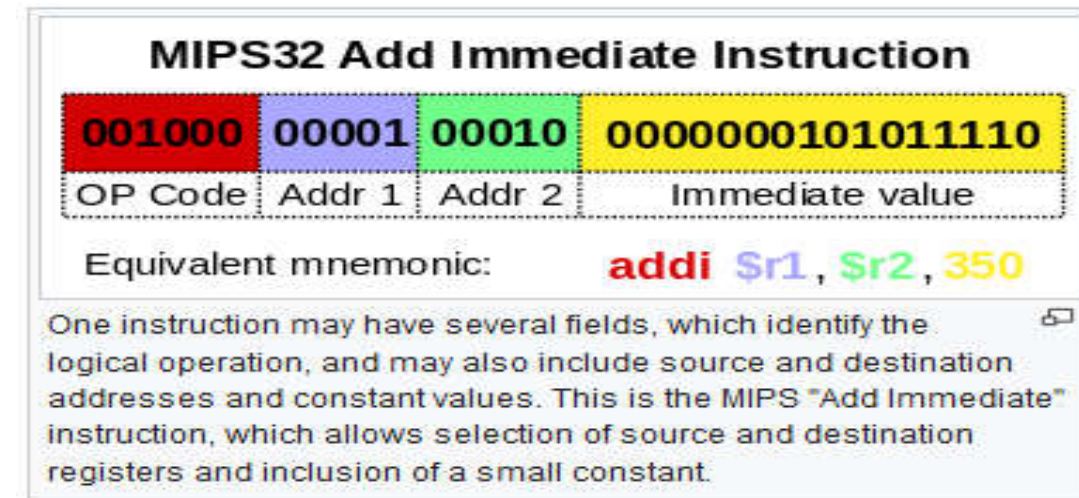
- Load/store data to and from a coprocessor, or exchanging with CPU registers.
- Perform coprocessor operations.

Parts of an Instruction

Setiap instruksi dapat mempunyai beberapa field yang mengidentifikasi **logical operation**, dan mungkin juga berisi **source dan destination addresses** dan **constant values**.

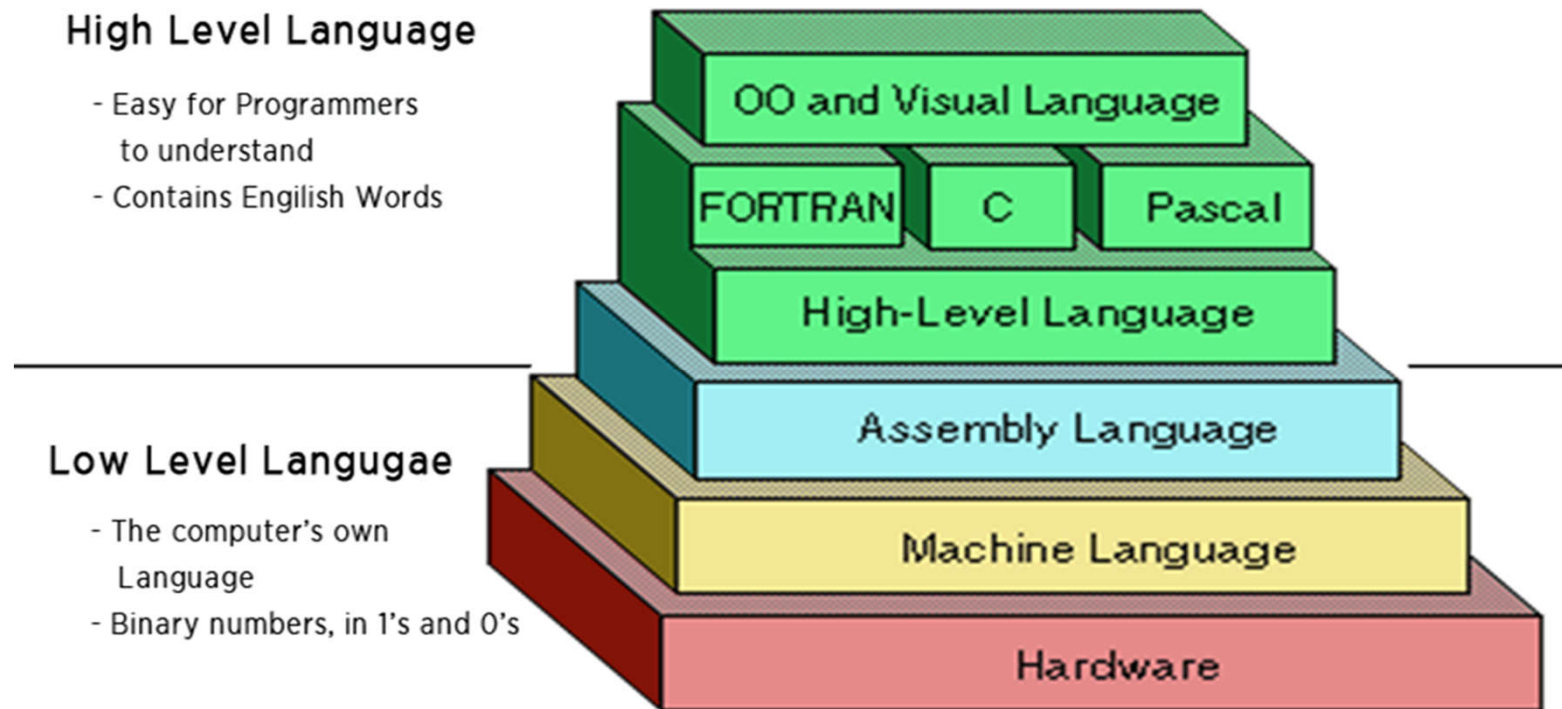
Contoh

MIPS "Add Immediate" instruction, which allows selection of source and destination registers and inclusion of a small constant.



Gambar : https://en.wikipedia.org/wiki/Instruction_set_architecture

Implementation of an Instruction



justcode.me

Point of View - ISA -

Performa komputer diukur dari seberapa cepat CPU dapat mengeksekusi job, yang dijadwalkan oleh Sistem Operasi.

ISA dibuat dengan maksud untuk dapat meminimalkan waktu proses dan memaksimalkan job yang dapat diproses pada satu waktu.

Acuan yang digunakan al:

- Model Instruksi /program : sederhana atau kompleks
- Siklus eksekusi : tiap job, global atau dapat dipecah2
- Waktu per siklus eksekusi : minimal?

Point of View - ISA -

Perbedaan cara pandang thd acuan tsb memunculkan 2 macam tipe arsitektur (single processor)

CISC (Complex Instruction Set Computing)

- Menekankan ke “instructions/program” dengan perintah yang “complex” (sekali perintah bisa selesai banyak)
- Mudah programmer (assembler), ringkas
- Satu siklus perintah butuh waktu selesai lama

RISC (Reduced Instruction Set Computing)

- Menekankan ke “instructions/program” dengan banyak perintah kecil sederhana, untuk satu job besar
- Perintah banyak, tidak ringkas
- Siklus perintah butuh waktu yang singkat, sehingga bisa “disisipi” perintah lain, meskipun job besar belum selesai

Instruction Set Architecture

There are two common types of architectures based on the instruction set, namely CISC (Computer Instruction Set Computer) and RISC (Reduced Instruction Set Computer).

CISC

Satu rangkaian instruksi mengerjakan complex job (banyak)

Misal : perintah mov

Contoh : x86 (AMD, Intel)

RISC

Setiap rangkaian instruksi mengerjakan simple job (per unit)

Misal : perintah add, sub

Arsitektur RISC memungkinkan **job dikerjakan secara pipeline**

Contoh : MIPS, ARM

ARM®



RISC

Gambar :

x86



CISC

RISC vs CISC

Perkalian Dua Bilangan dalam Memori
misal 5 x 7

Pendekatan CISC

- menyelesaikan task dengan rangkaian instruksi sesingkat mungkin, yang mungkin lebih dari 1 clock cycle
- Membuat processor yang mampu “mengerti” dan mengeksekusi rangkaian operasi singkat tersebut
- Processor menyiapkan perintah tertentu misal MULT, (ditempatkan di register)
- Ketika eksekusi, perintah ini meletakkan (load) 2 nilai tersebut (5 dan 7) di register yang berbeda misal A=5 dan B=7, mengalikan dan meletakkan hasil di register yang sesuai.

Contoh instruksi

MULT A,B

Pada processor sudah ditanamkan Perintah aritmetika berupa ADD dan SUB

RISC vs CISC

Perkalian Dua Bilangan dalam Memori
misal 5×7

Pendekatan CISC

- MULT adalah complex instruction, high level language, programmer (compiler) tidak perlu mendeskripsikan fungsi loading dan storing data
- Kebutuhan space register kecil
- Butuh space register (transistor) lain untuk menyimpan complex instruction
- Harus membuat complex instruction langsung ditanam pada hardware

RISC vs CISC

Perkalian Dua Bilangan dalam Memori
misal 5×7

Pendekatan RISC

- menyelesaikan task dengan **instruksi simple** yang dapat diselesaikan dalam **1 clock cycle**
- Perintah MULT seperti di CISC, dibagi menjadi 3 perintah sederhana,
 - LOAD : memindah data dari memory ke register
 - PROD : menemukan produk dari dua operan, pada register
 - STORE : memindah data dari register ke memory bank
- misal $A=5$ dan $B=7$, untuk mengalikan dan meletakkan hasil di register yang sesuai.

Contoh instruksi

```
LOAD A,5  
LOAD B,7  
PROD A,B  
STORE 35 , A
```

RISC vs CISC

Perkalian Dua Bilangan dalam Memori
misal 5 x 7

Pendekatan RISC

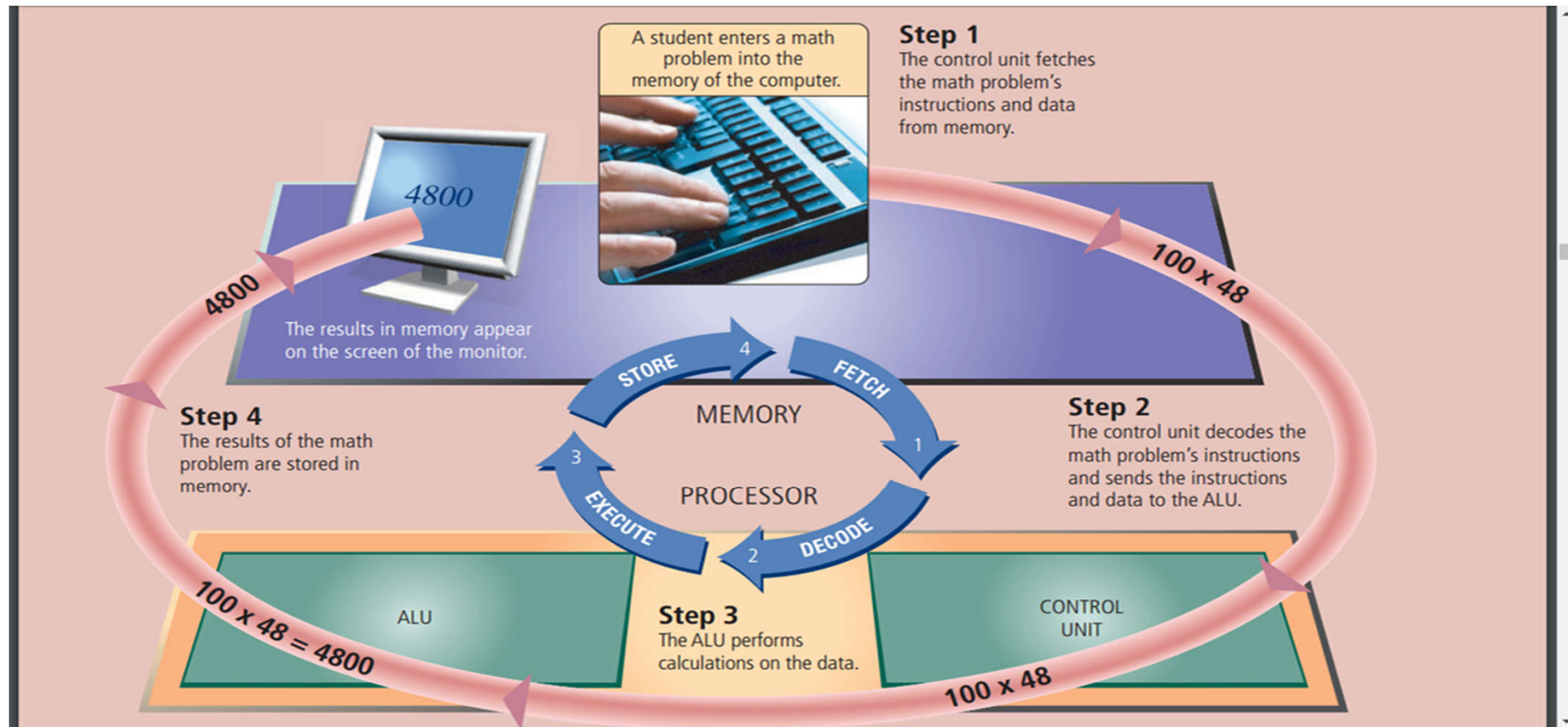
- Compiler/software perlu mengkonversi perintah menjadi kode yang dimengerti processor
- Kebutuhan space register lebih banyak
- Tidak butuh space register (transistor) lain untuk perintah, semua dapat dimaksimalkan untuk kebutuhan penyimpanan data
- Karena setiap item perintah dapat diselesaikan dalam 1 clock cycle, memungkinkan untuk **pipelining**

RISC vs CISC

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
Small code sizes, high cycles per second	Low cycles per second, large code sizes
Transistors used for storing complex instructions	Spends more transistors on memory registers

Tabel : <https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risciscisc/>

Computer Execution Instruction cycle



Computer Execution Instruction cycle

IF	ID	EX	MEM	WB
IF	: Instruction fetch			
ID	: Instruction Decode			
EX	: Execute			
MEM	: Memory Access			
WB	: Write Back			

IF

- Ambil instruksi berikutnya dari memory

ID

- Memastikan perintah apa yg harus dikerjakan
- Mengambil nilai dari register tertentu (sesuai alamat) sebelum dikodekan

EX

- Pada referensi memory, tambahkan *Up Base* dan *Offset*
- Pada instruksi arimatika, mengerjakan perintah matematika

MEM

- Jika *Load* atau *Store*, akses memory
- Jika percabangan, ganti *ProgramCounter* dengan *destination address*

WB

- Simpan hasil pada alamat register tertentu

Computer Execution Instruction cycle

IF	ID	EX	MEM	WB
IF	: Instruction fetch			
ID	: Instruction Decode			
EX	: Execute			
MEM	: Memory Access			
WB	: Write Back			

Contoh berikut adalah eksekusi 5 job dan 2 paralel, dengan model eksekusi diatas perhatikan perbedaan waktu penyelesaian job

Contoh :

Jika 1 job memerlukan 5 tahap instruksi (IF,ID,EX,MEM,WB)

Berikut adalah model eksekusi perintah oleh

- Scalar
- Pipeline
- SuperScalar
- SuperPipeline

Superscalar dan Superpipeline mampu mengerjakan job "secara paralel"

Jumlah paralel bisa lebih dari 2

Computer Execution Instruction cycle

SCALAR processor

Instruction No	SCALAR STAGE																								
1	IF	ID	EX	MEM	WB																				
2						IF	ID	EX	MEM	WB															
3											IF	ID	EX	MEM	WB										
4																IF	ID	EX	MEM	WB					
5																					IF	ID	EX	MEM	WB
Clock Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

- contoh 1 job desk menggunakan SCALAR
- 1 clock 1 instruksi
 - 1 job per sesi

Computer Execution Instruction cycle

PIPELINE processor

Instruction No	PIPELINE STAGE								
1	IF	ID	EX	MEM	WB				
2		IF	ID	EX	MEM	WB			
3			IF	ID	EX	MEM	WB		
4				IF	ID	EX	MEM	WB	
5					IF	ID	EX	MEM	WB
Clock Cycle	1	2	3	4	5	6	7	8	9

- 1 tahapan pipeline

Computer Execution Instruction cycle

SUPERSCALAR processor

Instruction No	SUPER SCALAR STAGE						
1	IF	ID	EX	MEM	WB		
2	IF	ID	EX	MEM	WB		
3		IF	ID	EX	MEM	WB	
4		IF	ID	EX	MEM	WB	
5			IF	ID	EX	MEM	WB
Clock Cycle	1	2	3	4	5	6	7

- 1 tahapan pipeline

Computer Execution Instruction cycle

SUPERPIPELINE processor

Instruction No	SUPER PIPELINE STAGE									
	IF	ID	EX	MEM	WB					
1	IF	ID	EX	MEM	WB					
2		IF	ID	EX	MEM	WB				
3			IF	ID	EX	MEM	WB			
4				IF	ID	EX	MEM	WB		
5					IF	ID	EX	MEM	WB	
Clock Cycle	1	2	3	4	5					

- 1 clock 2 instruksi
- 2 tahapan pipeline

Key term

branch prediction condition code delayed branch	flag instruction cycle instruction pipeline	instruction prefetch program status word (PSW)
complex instruction set computer (CISC) delayed branch delayed load	high-level language (HLL) reduced instruction set computer (RISC)	register file register window SPARC

Referensi

UTAMA

- ❑ William Stalling, Computer Organization and organization 8th edition, Pearson Education, Inc, Pearson Prentice Hall, 2010
- ❑ Andrew S. Tanenbaum, Structured Computer Organization 4th Edition Pearson Prentice Hall, 2001
- ❑ Mostafa Abd-El-Barr- Hesham El-Rewini, Fundamentals Of Computer Organization And Architecture, John Wiley & Sons, Inc, 2005

TAMBAHAN

- ❑ <http://www.computerhistory.org>
- ❑ <https://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading04.htm>
- ❑ <https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/riscisc/>
- ❑ https://www.electronics-tutorials.ws/binary/bin_2.html
- ❑ <http://www.ict.griffith.edu.au/~johnt/1004ICT/lectures/>
- ❑ <https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/riscisc/>