PROGRAM STUDI
TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO

Instruction Set Architecture dan Desain

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

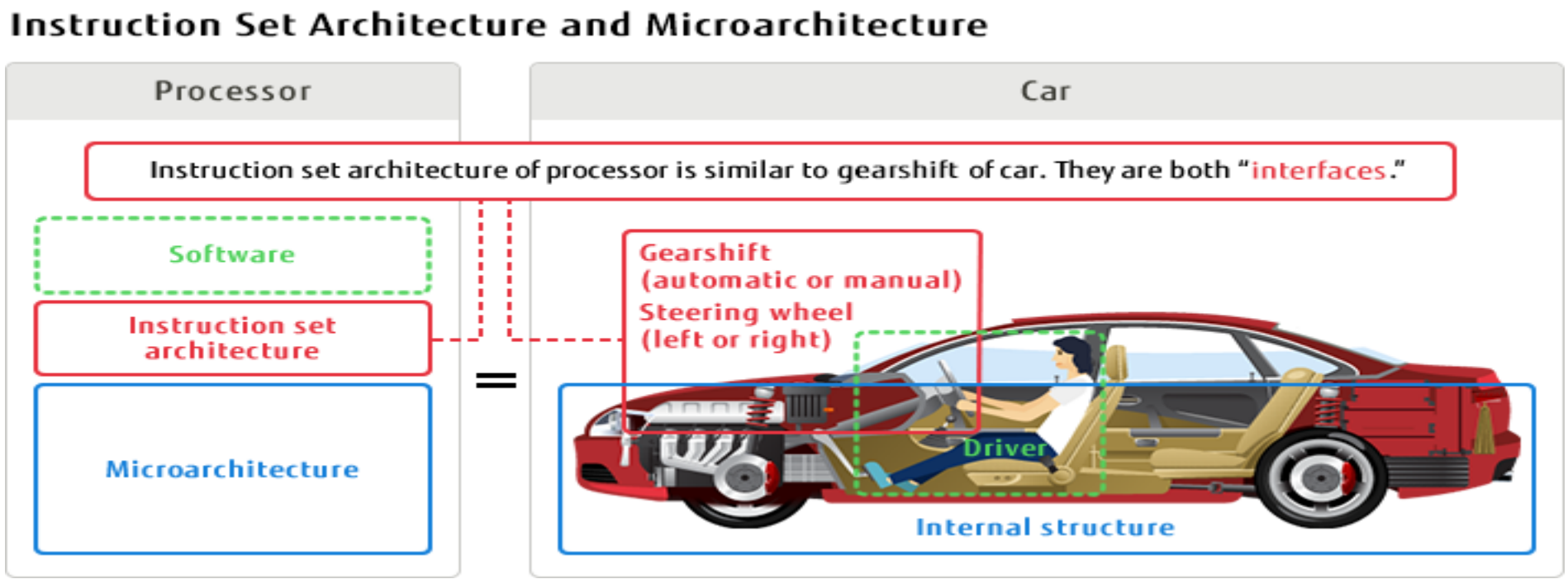
Tim pengampu

Sistem Komputer, Komunikasi dan Keamanan Data

T.A. 2020

Instruction Set Architecture

Bagaimana berkomunikasi dengan komputer?

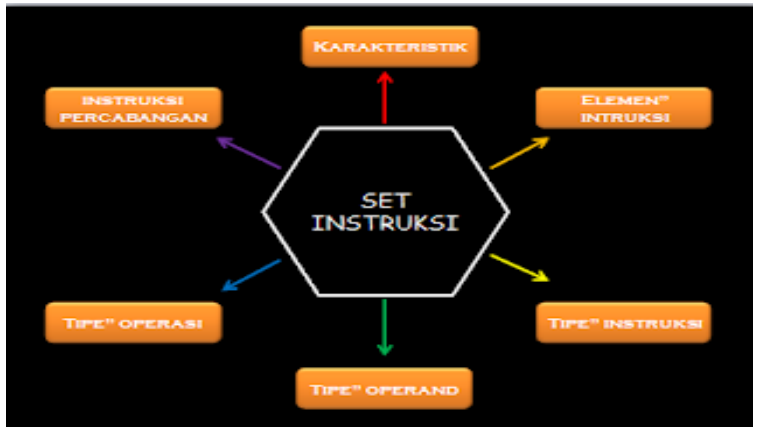


Gambar : http://journal.jp.fujitsu.com

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 I/O 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.

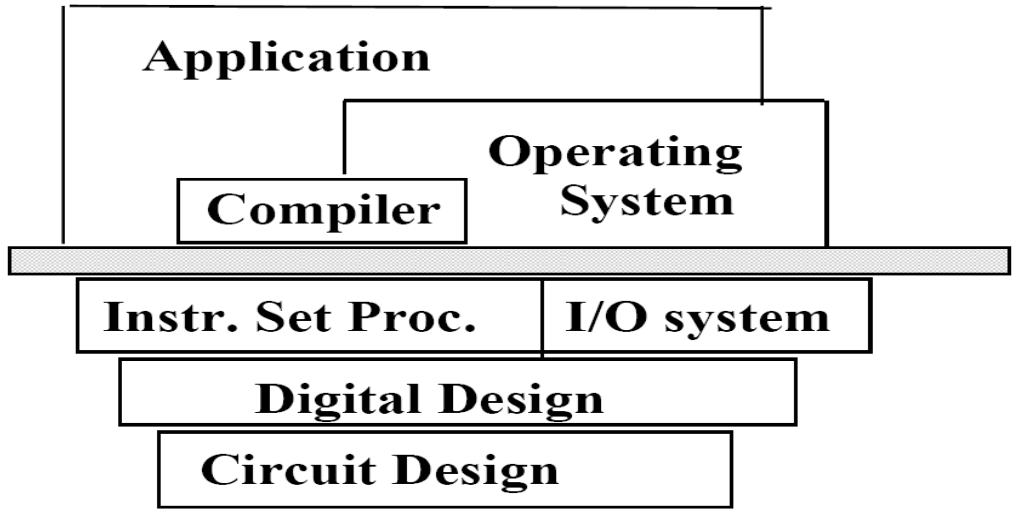
- Setiap vendor prosesor secara mandiri mendefinisikan struktur prosesor internal mereka sendiri.
- Contoh : Fujitsu dan Oracle mendesain struktur internal dari prosesor sendiri, secara mandiri, tetapi prosesor mereka menggunakan ISA SPARC yang sama.
- Struktur internal prosesor (microarchitecture),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

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:

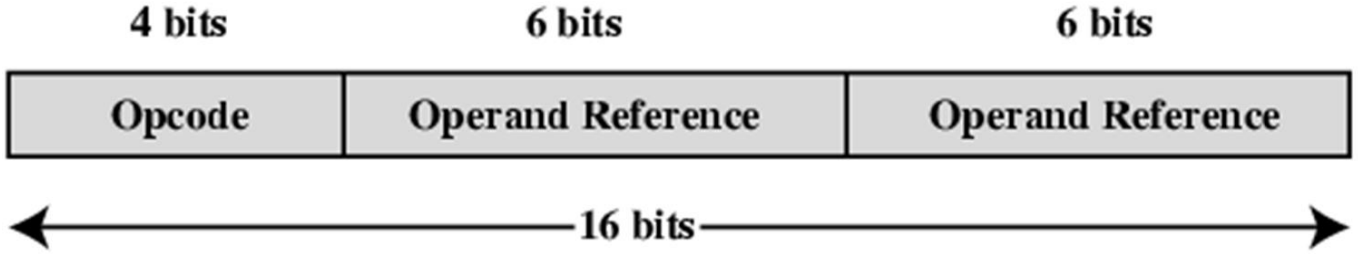
- register khusus untuk aritmatika, addressing, atau control functions
- lokasi memori atau offset tertentu
- mode addressing tertentu yang digunakan untuk menafsirkan (interpret) operand



Gambar : allansnavely-L2isa.pdf

Instruction Set Architecture (ISA)

- ISA adalah Kumpulan perintah (fungsi) yang dapat dieksekusi CPU
- ISA membentuk suatu operasi sistemik yang meliputi berbagai macam fungsi



Elemen ISA

1. OpCode : Operation Code
   Spesifikasi operasi yang akan dilakukan / pengkodean operasi
2. Source *Operand* Reference
   Input bisa lebih dari satu sumber
3. Result *Operand* Reference
   Nilai hasil dari eksekusi operasi
4. Next Instruction Reference
   Posisi Instruksi berikutnya yang harus diambil dan dieksekusi

Instruction Set Architecture (ISA)

OpCode direpresentasikan dengan singkatan yang menggambarkan Fungsi Operasi untuk CPU, disebut mnemonic.

Contoh mnemonic adalah :

ADD	= Add	penambahan
SUB	= Substract	pengurangan
LOAD	= Load	Memuat data ke memori

Contoh
ADD X,Y
tambahkan nilai Y ke nilai register X, (dan simpan hasilnya di register X)

Programmer dapat menempatkan Operand X dan Y di lokasi memory tertentu, Misal
X direferensikan ke alamat 111
Y direferensikan ke alamat 222

Representasi program
LOAD sebuah nilai register dengan alamat 111
ADD dengan isi alamat register 222
Simpan hasilnya di alamat 111

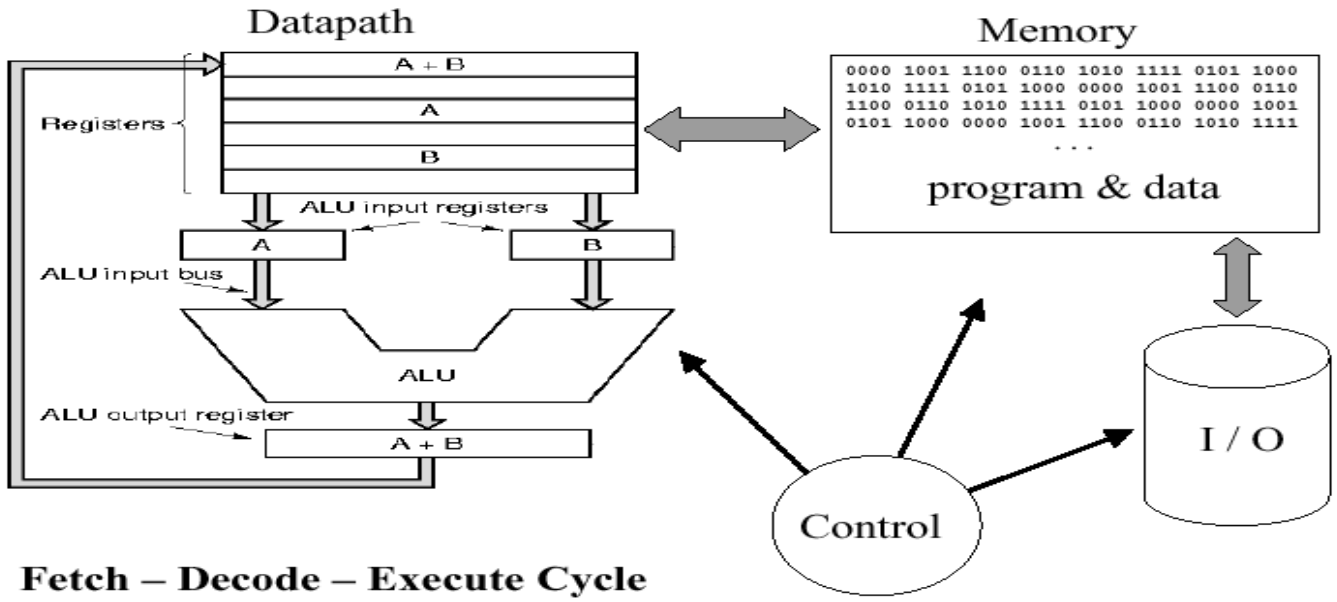
Memory Location

Dari sumbernya posisi Operand akan berada pada area berikut

- Memory (utama/virtual)
- Register CPU
- I/O device

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 periferal seperti disk atau keyboard).



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*

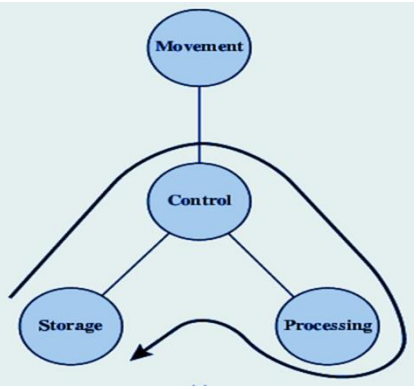
Contoh, memory 256Mb dapat menampung sekitar : $256x2^{20} = 2^{28}$ byte

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

Type Instruction

Contoh operasi yang umum untuk banyak set instruksi termasuk:

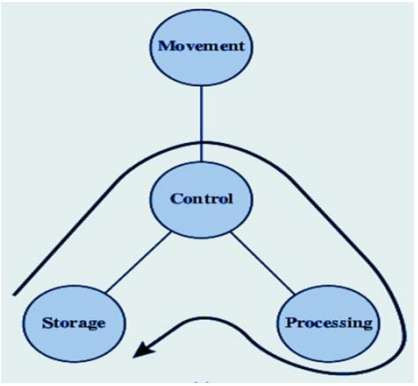
1. Pengolahan Data (Data Processing)
2. Perpindahan Data (Data Movement)
3. Penyimpanan Data (Data Storage)
4. Kontrol Aliran Operasi (Control)



1. Pengolahan Data (Data Processing)

Meliputi operasi Logic dan Aritmatika

- *Add*, *subtract*, *multiply*, atau *divide* nilai dari 2 regsiter 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.



2. Perpindahan Data (Data Movement)

berisi instruksi perpindahan data antar register maupun modul I/O

Dibutuhkan instruksi untuk memindahkan Operand yang diperlukan untuk diproses CPU (*Coprocessor instructions*)

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

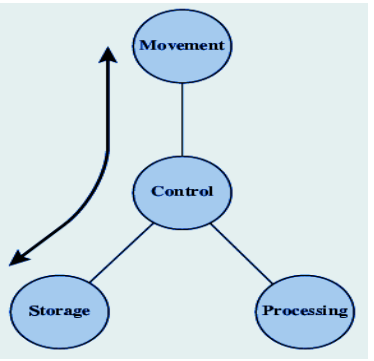
Type Instruction

Penyimpanan Data (Data Storage)

Sebagai *Data handling and memory operations*

berisi instruksi penyimpanan ke memori, meskipun sementara, untuk digunakan proses berikutnya atau tampungan untuk ditampilkan di Output device

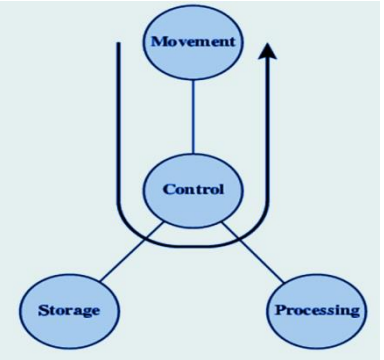
- Memberikan nilai konstanta ke register
- Copy data dari lokasi memory ke register atau sebaliknya (load and store operations).
- *Read* and *write* data dari hardware devices.



Kontrol Aliran Operasi (Control)

berisi instruksi untuk mengontrol operasi dan pencabangan. Berguna sebagai *status* suatu operasi dan atau pencabangan ke set instruksi lain

- *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.



Transfer Data

Instruksi tranfer data harus menetapkan :
Lokasi operand sumber
Lokasi operand tujuan
Panjang data yang akan dipindahkan
Mode pengalamatannya

Apabila sebuah atau kedua operand berada di dalam memori, maka CPU harus melakukan sebagian atau seluruh tindakan berikut :

1. Menghitung alamat memori, yang didasarkan pada mode alamatnya.
2. Jika alamat mengacu pada virtual memori harus dicari alamat memori sebenarnya.
3. Menentukan apakah alamat berada dalam cache memori.
4. Bila di cache tidak ada dikeluarkan perintah ke modul memori

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

Jumlah Alamat (register) Instruksi

- Jumlah register yang digunakan tergantung tipe masing-masing CPU
- Format instruksi ada yang mengunakan alamat register 3,2,1,atau 0
- Format yang umum digunakan adalah 2 register alamat dalam suatu operasi
- Desain CPU sekarang sudah menggunakan 3 register alamat, terutama dalam dalam MIPS (Million Instruction Per Second)

Jumlah Alamat (register) Instruksi

Alamat Instruksi Sedikit (1 atau 0)

- Instruksi lebih sederhana dan pendek, tetapi sulit diimplementasikan di pemrograman (programmer)
- Desain CPU sederhana
- Jumlah bit dan referensi instruksi *sedikit*, maka proses (fetch-decode-execute) lebih cepat
- Jumlah instruksi per program lebih banyak (baris coding lebih banyak)
- Butuh Register banyak, operasi antar register lambat (bolak-balik ambil data)

Alamat Instruksi Banyak (2 atau 3)

- Instruksi lebih kompleks dan panjang, tetapi mudah diimplementasikan di pemrograman (programmer)
- Desain CPU lebih kompleks
- Jumlah bit dan referensi instruksi *Banyak* , maka proses (fetch-decode-execute) lebih lama
- Jumlah instruksi per program lebih sedikit (baris coding lebih sedikit)
- Butuh Register banyak, operasi antar register cepat

Contoh

3 Alamat

Operand 1, Operand 2, Hasil

ADD a,b,c : a = b + c;

Ada kemungkinan, alamat ke-4 (berikutnya) adalah *next instruction*

Butuh *word* yang panjang untuk menyimpan semua

2 Alamat

Salah satu alamat merangkap sebagai operand dan Hasil

SUB a,b : a = a - b

1 Alamat

Alamat kedua implisit

Biasanya berupa register (accumulator/AC)

Umum pada CPU generasi awal

LOAD a : AC = a

SUB b : AC = AC-b

STOR c : c = AC

0 Alamat

Semua alamat Implisit. Menggunakan *stack*

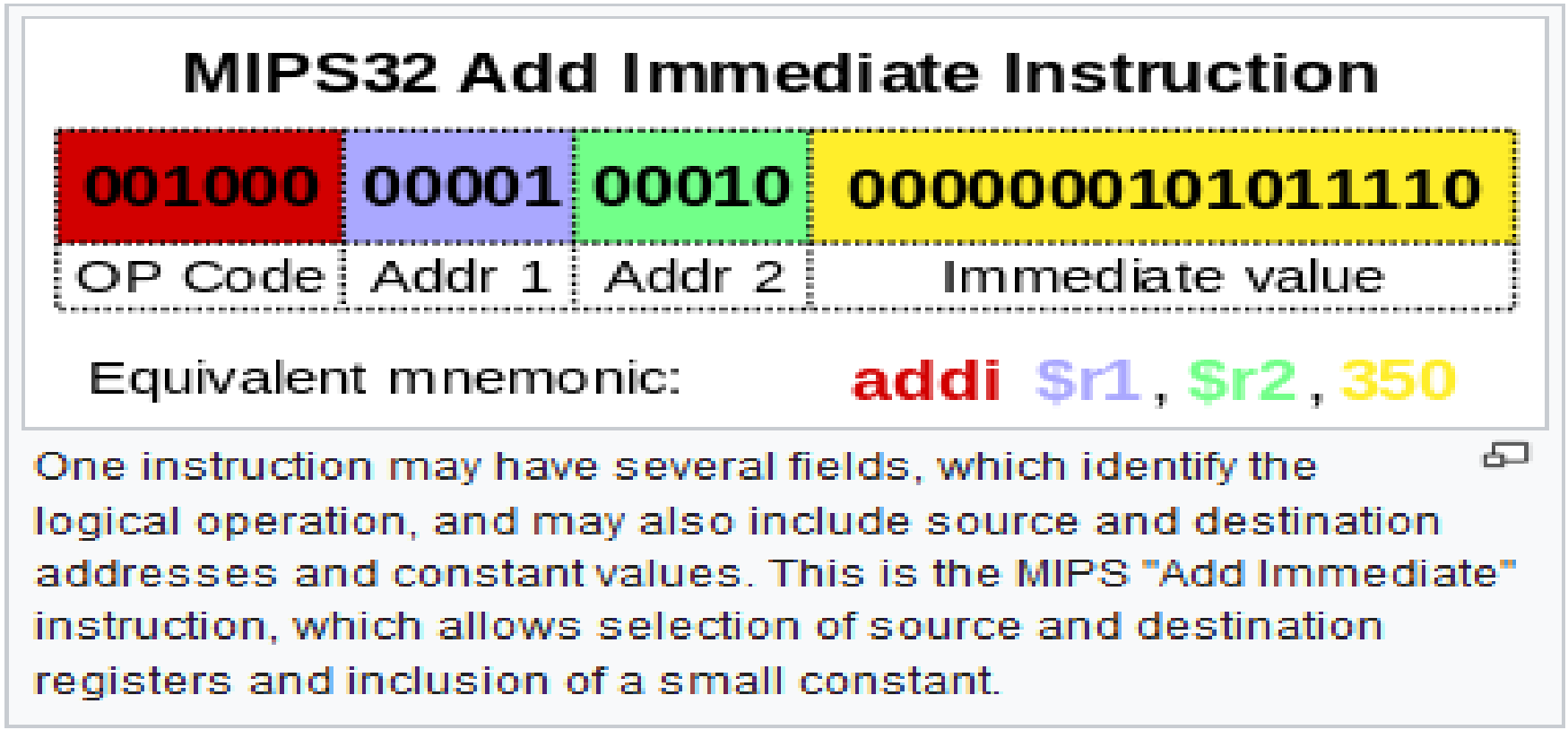
PUSH a

PUSH b

ADD

POP c

c = a + b

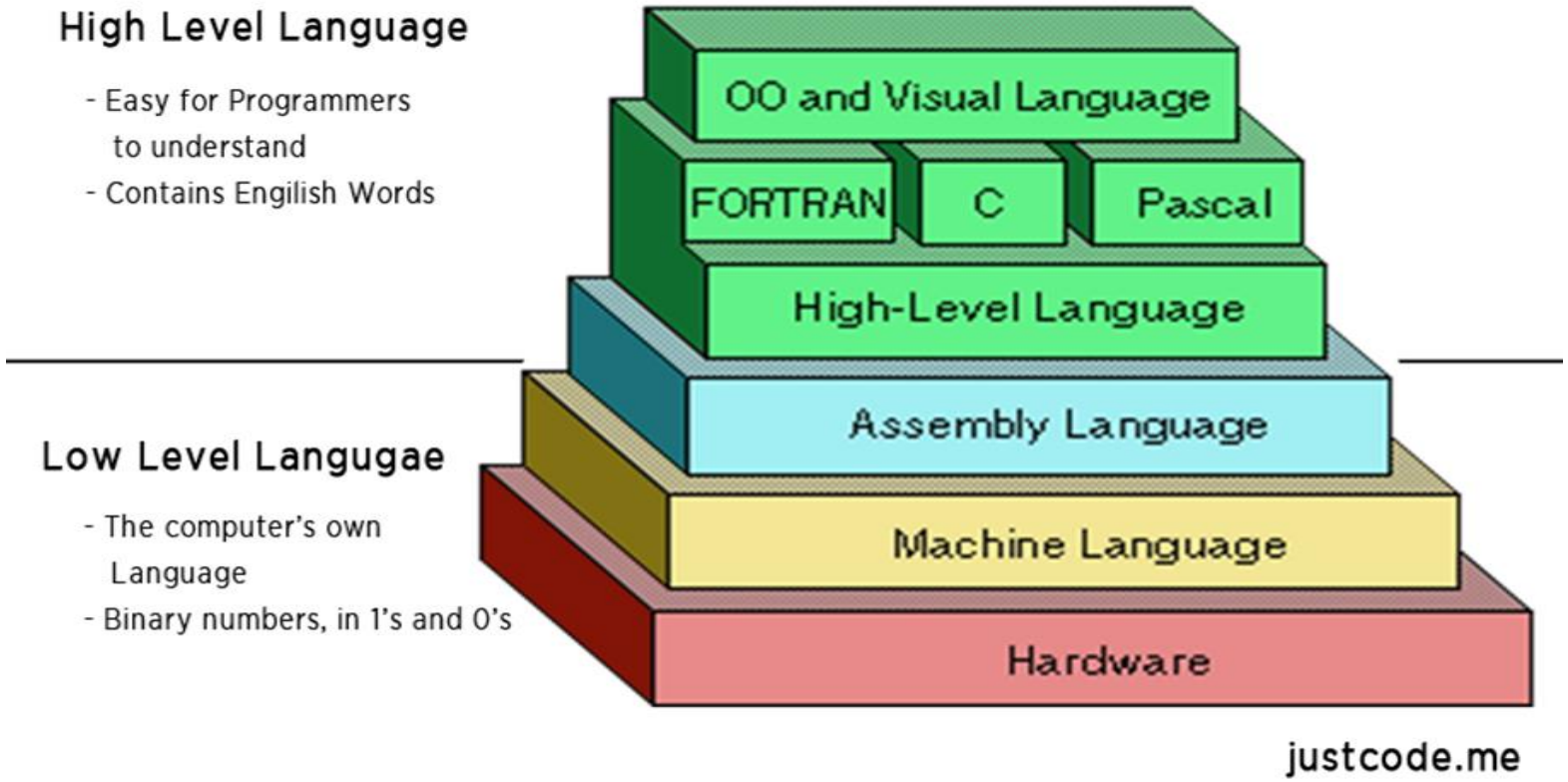


Gambar : https://en.wikipedia.org/wiki/Instruction_set_architecture

Contoh

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

Implementation of an Instruction



ISA yang diimplementasikan dalam bentuk perangkat keras
https://id.wikipedia.org/wiki/Set_instruksi

- Alpha AXP (DEC Alpha)
- ARM (Acorn RISC Machine)
  (Advanced RISC Machine now ARM Ltd)
- IA-64 (Itanium/Itanium 2)
- MIPS
- Motorola 68k
- PA-RISC (HP Precision Architecture)
- IBM POWER
- IBM PowerPC
- SPARC
- SuperH (Hitachi)
- System/360
- Tricore (Infineon)
- Transputer (STMicroelectronics)
- VAX (Digital Equipment Corporation)
- x86 (IA-32, Pentium, Athlon) (AMD64, EM64T)

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 komplek
- 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 programer (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

Gambar :

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

- 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 x 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

Contoh instruksi

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

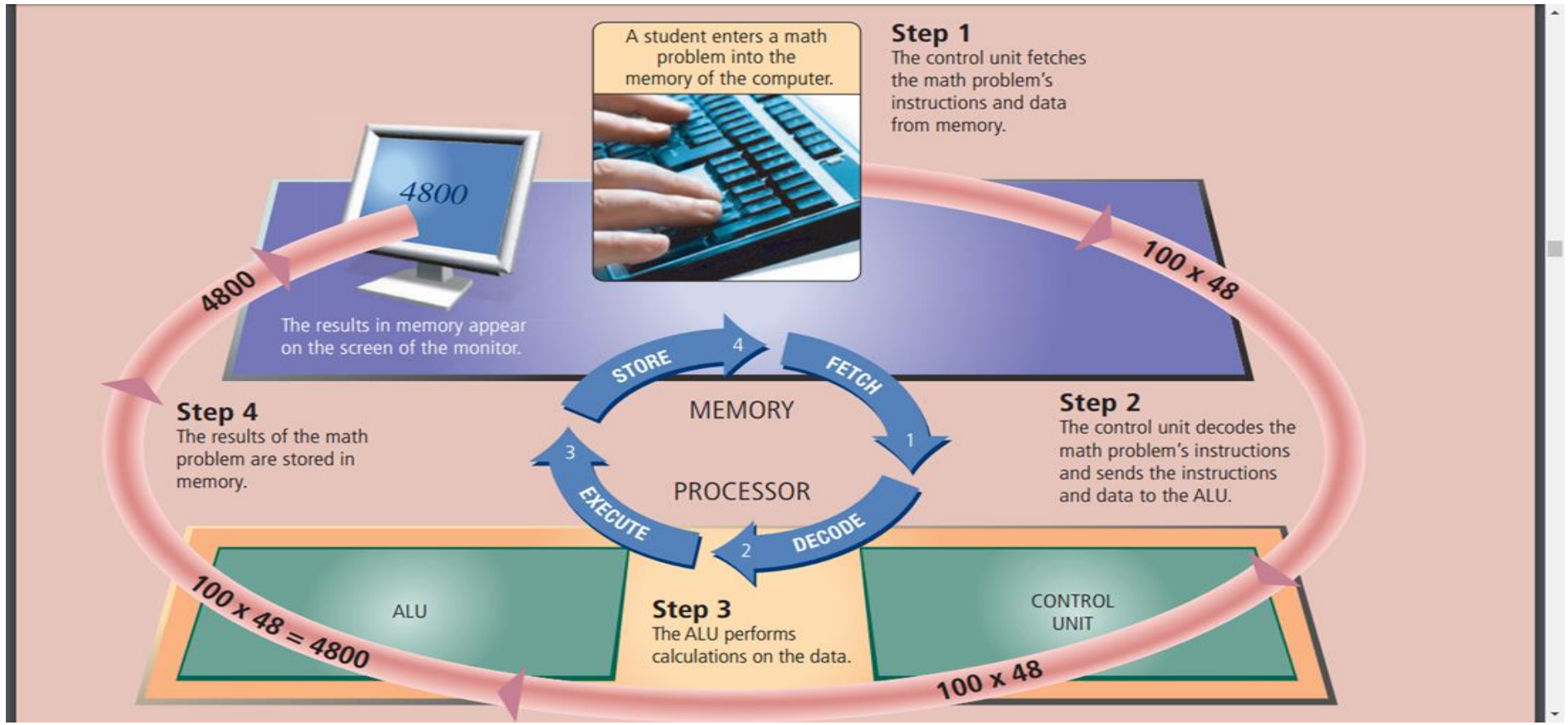
- Compiler/software perlu mengkonversi perintah menjadi kode yang dimengerti processor
- Kebutuhan space register lebih banyak
- Tidak butuh space register (transistor) lain untuk perintah, semua dapat dimaksimalkan untuk kebutuhan penyimpanan data
- Karena setiap item perintah dapat diselesaikan dalam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

RISC vs CISC

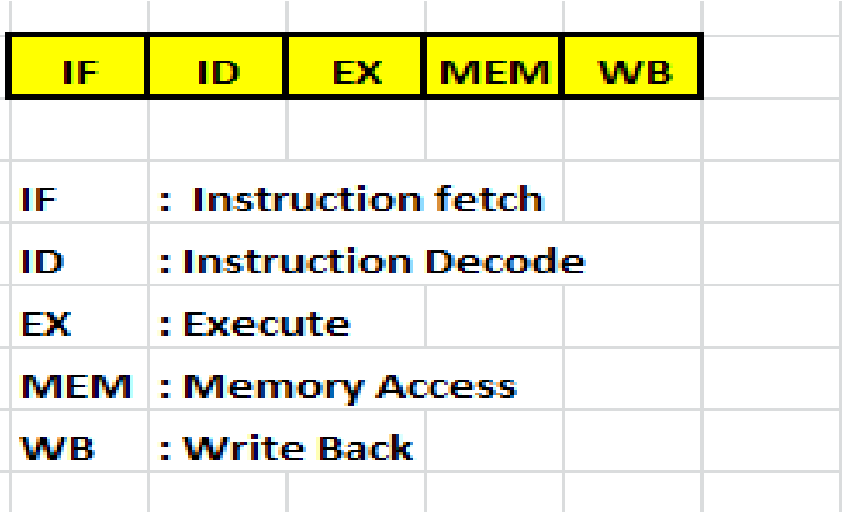
CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
Small code sizes, high cycles per 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/risccisc/

Computer Execution Instruction cycle



Computer Execution Instruction cycle



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 *ProgamCounter* 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 :
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

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

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 : 5 job dieksekusi dg 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

Contoh : 5 job dieksekusi dg PIPELINE

- 1 clock 1 instruksi
- 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

Contoh : 5 job dieksekusi SUPERSCALAR dengan 2 paralelism

- 1 clock 2 instruksi
- 1 tahapan pipeline

Computer Execution Instruction cycle

SUPERPIPELINE processor

Instruction No	SUPER 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	

Contoh : 5 job dieksekusi SUPERPIPELINE dengan 2 paralelism

- 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/risccisc/
- 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/risccisc/