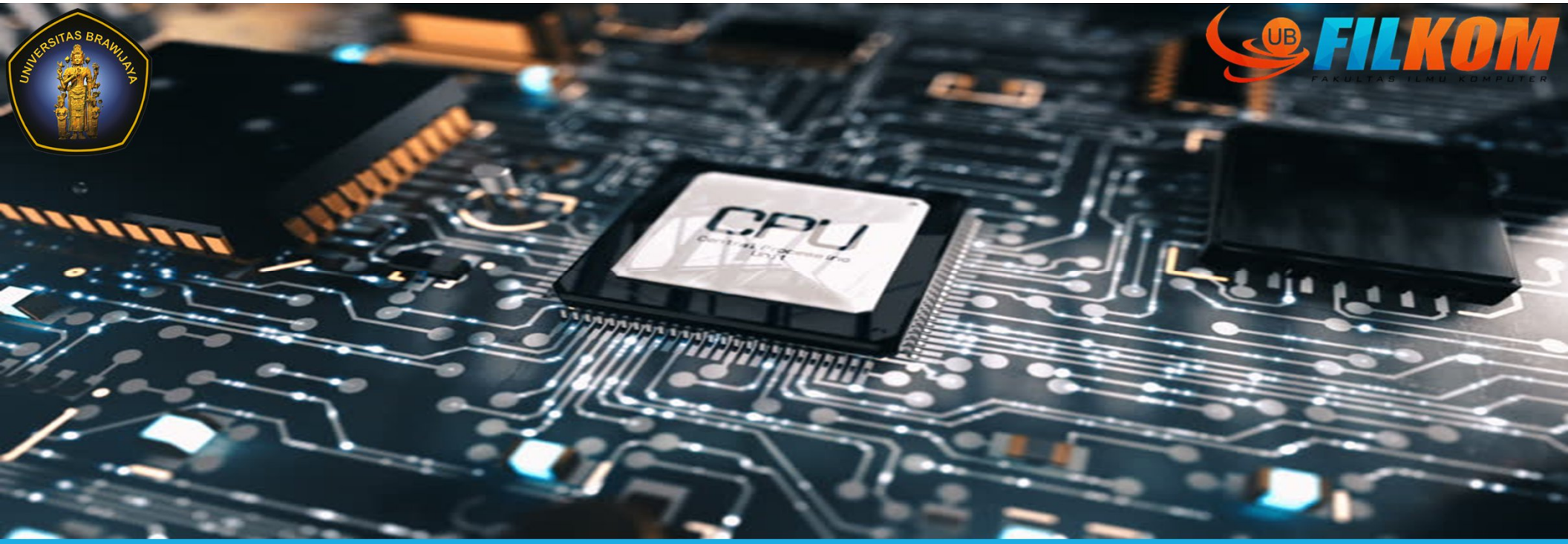


Arsitektur dan Organisasi Komputer

COM 60011

Topik #11 – Instruction Sets: Addressing Modes and Formats

Tujuan

CPMK Mahasiswa mampu menjabarkan Instruction Set Architecture (ISA) yang ada dan digunakan pada komputer.

Sub CPMK : Mahasiswa mampu menjelaskan tentang Instruction Set Architecture (ISA) dari segi karakteristik dan pengalamatannya yang digunakan pada komputer.

Materi Terkait:

- Mode Pengalamatan
- Format Instruksi
- Pengalamatan dan Instruksi pada x86
- Pengalamatan dan Instruksi pada ARM

Instruction Sets: Addressing Modes and Formats

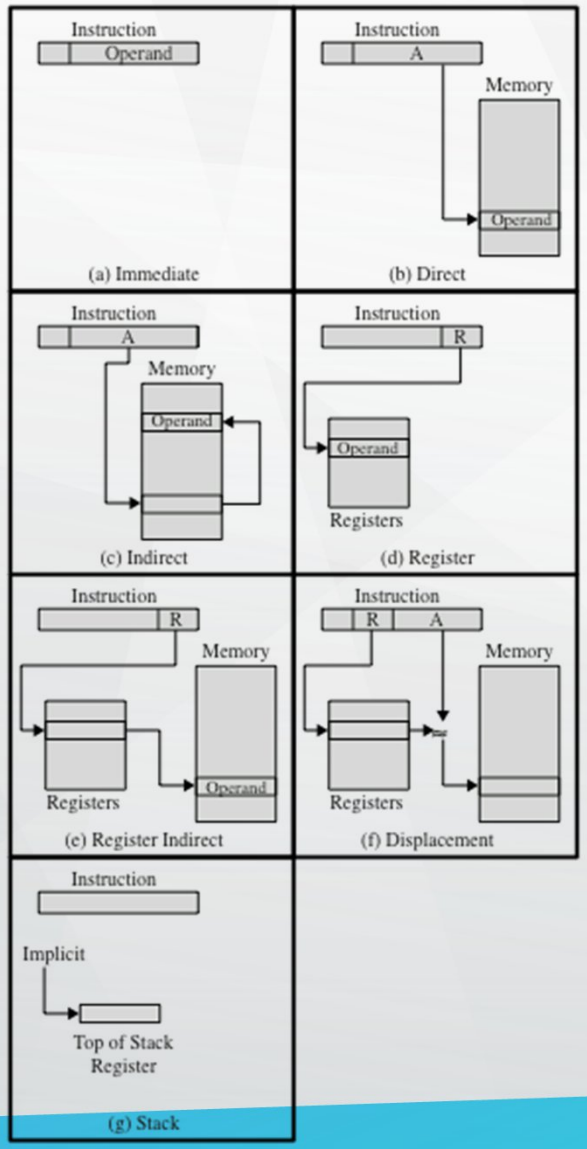
Pendahuluan

Suatu variasi mode pengalamatan (addressing mode) dapat digunakan untuk menentukan suatu alamat tempat untuk dimanaoperand akan di fetch.

Beberapa teknik dapat meningkatkan kecepatan pelaksanaan instruksi dengan menurunkan jumlah referensi pada memori utama dan meningkatkan jumlah referensipada register kecepatan tinggi.

Mode pengalamatan ini menjabarkan suatu aturan untuk menginterpretasikan atau memodifikasi field alamat dari instruksi sebelum operand direferensikan.

Addressing Modes



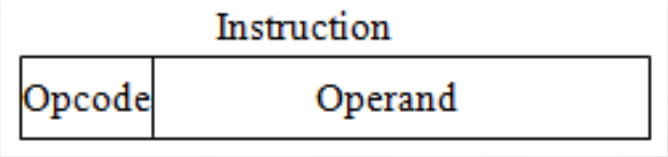
Basic Addressing Modes

Mode	Algorithm	Principal Advantage	Principal Disadvantage
Immediate	Operand = A	No memory reference	Limited operand magnitude
Direct	EA = A	Simple	Limited address space
Indirect	EA = (A)	Large address space	Multiple memory references
Register	EA = R	No memory reference	Limited address space
Register indirect	EA = (R)	Large address space	Extra memory reference
Displacement	EA = A + (R)	Flexibility	Complexity
Stack	EA = top of stack	No memory reference	Limited applicability

Immediate Addressing

Bentuk pengalamatan yang paling sederhana, di mana operand langsung ada pada instruksi Format intruksi.

- Operand benar-benar ada dalam instruksi atau bagian dari instruksi (operand sama dengan field alamat).
- Umumnya bilangan akan di simpan dalam bentuk komplemen dua.
- Bit paling kiri sebagai bit tanda.
- Operand = address field
- Contoh : ADD 5
  - Tambah nilai 5 ke isi akumulator
  - 5 adalah operand
- Diagram Immediate addressing



Isi address field langsung berupa operand

- Keuntungan :
  - Tidak ada referensi memori selain dari instruksi yang diperlukan untuk memperoleh operand.
  - Menghemat siklus instruksi sehingga proses keseluruhan akan cepat.
- Kerugian:
  - Ukuran bilangan (operand) dibatasi oleh ukuran address field.

Direct Addressing

- Address field contains the effective address of the operand
- Effective address (EA) = address field (A)
- Was common in earlier generations of computers
- Requires only one memory reference and no special calculation
- Limitation is that it provides only a limited address space

Direct Addressing

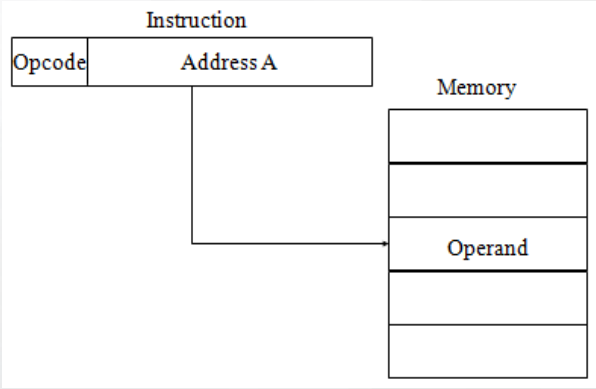
- Kelebihan :
  - Field alamat berisi efektif address sebuah operand
  - Teknik ini banyak digunakan pada komputer lama dan komputer kecil.
  - Hanya memerlukan sebuah referensi memory dan tidak memerlukan kalkulasi khusus.

- Kelemahan :
  - Keterbatasan field alamat karena panjang field alamat biasanya lebih kecil dibandingkan panjang word.

- Adress field berisi alamat dari operand

- Effective address (EA) = address field (A)
  - A = Addressing · isi bit suatu field alamat dalam instruksi
  - EA = Effective Addressing · alamat aktual sebuah lokasi yang berisi

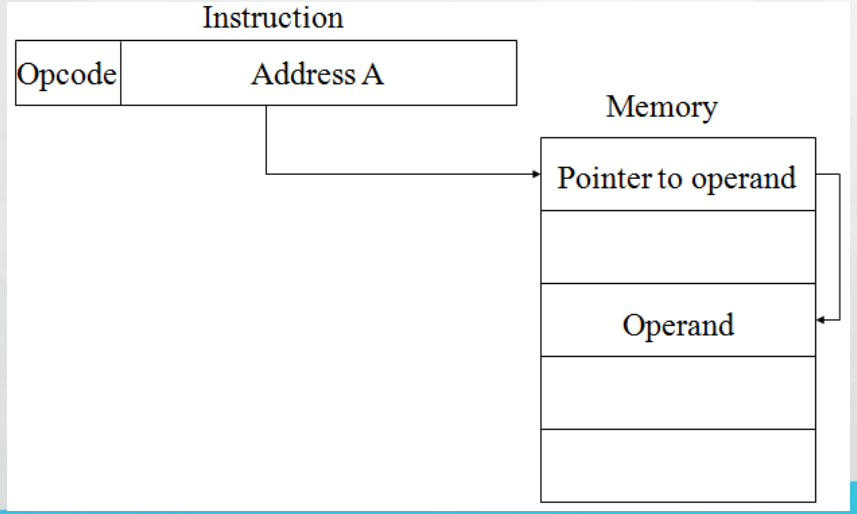
- Contoh : ADD A
  - Cari di memory pada alamat A untuk operand
  - Tambahkan isi yang ada pada alamat A dengan nilai di register accumulator dan simpan hasilnya di register accumulator.



Indirect Addressing

Merupakan mode pengalamatan operand dimana area alamat (address field) berisi alamat dari suatu alamat yang akan menunjukkan alamat dari suatu nilai yang akan diproses. Field alamat mengacu pada alamat word didalam memory, yang pada gilirannya akan berisi alamat operand yang panjang.

Contoh: ADD (A)
Keterangan: Tambahkan isi dari cell yang alamatnya ditunjukkan oleh isi yang terdapat pada A dengan nilai yang ada di register accumulator dan simpan hasilnya di register accumulator.



Indirect Addressing

- Karakteristik:
  - Memerlukan space address yang besar.
  - 2n dimana n adalah panjang word
  - Dapat dibuat nested (bersarang), multilevel dan cascade (bertumpuk).

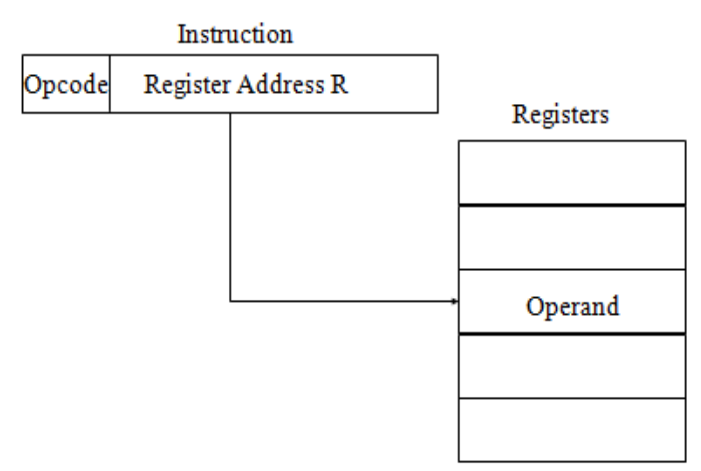
Contoh : EA = (((A)))

  - Pengaksesan memory yang multiple untuk mendapatkan operand sehingga mengakibatkan proses mode ini agak lebih lambat

- Keuntungan:
  - Jumlah lokasi memori yang dapat diacu lebih banyak.
  - Hanya membutuhkan satu referensi memori.
  - Tidak memerlukan kalkulasi alamat yang khusus.
  - Ruang bagi alamat menjadi besar sehingga semakin banyak alamat yang dapat referensi
- Kerugian:
  - Memerlukan 2 referensi memori.
  - Jumlah lokasi yang diacu dibatasi oleh ukuran field alamat.
  - Diperlukan referensi memori ganda dalam satu fetch sehingga memperlambat proses operasi

Register Addressing

- Operand ada di penamaan register di address field
- EA = R
- Nomor dari register dibatasi
- Alamat yang sangat kecil membutuhkan sedikit instruksi dan instruksi fetch yang cepat
- Tidak ada memori akses
- Eksekusi yang sangat cepat
- Sangat sedikit tempat dari alamat yang disediakan
- Multiple register membutuhan performance
- Diagram register addressing



Register Addressing

- Keuntungan :
  - Metode pengalamatan register mirip dengan mode pengalamatan langsung.
  - Perbedaannya terletak pada field alamat yang mengacu pada register, bukan pada memory utama.
  - Field yang mereferensi register memiliki panjang 3 atau 4 bit, sehingga dapat mereferensi 8 atau 16 register general
  - Pengambilan operand dari register lebih cepat dari pengambilan operand dari memori.
  - Jumlah register jauh lebih sedikit dibanding dengan jumlah lokasi memori, sehingga jumlah bit alamat yang dibutuhkan lebih sedikit akibatnya ukuran address bisa lebih kecil.
  - Diperlukan field alamat berukuran kecil dalam instruksi dan tidak diperlukan referensi memori
  - Akses ke regster lebih cepat daripada akses ke memori, sehingga proses eksekusi akan lebih cepat
- Kerugian :
  - Jumlah register sangat terbatas, sehingga penggunaannya harus benar-benar efisien.
  - Ruang alamat menjadi terbatas

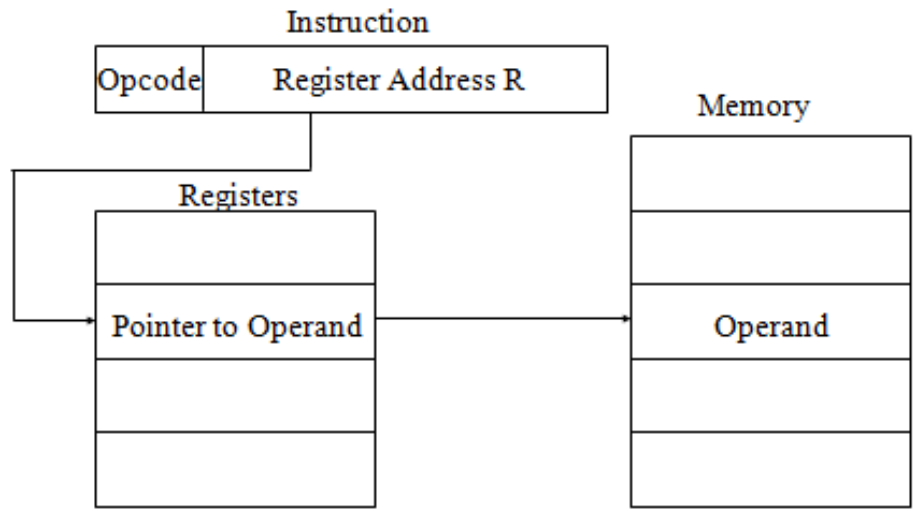
Register Indirect Addressing

- Metode pengalamatan register tidak langsung mirip dengan mode pengalamatan tidak langsung
- Perbedaannya adalah field alamat mengacu pada alamat register.
- Letak operand berada pada memori yang dituju oleh isi register
- Keuntungan dan keterbatasan pengalamatan register tidak langsung pada dasarnya sama dengan pengalamatan tidak langsung
- Keterbatasan field alamat diatasi dengan pengaksesan memori yang tidak langsung sehingga alamat yang dapat direferensi makin banyak
- Dalam satu siklus pengambilan dan penyimpanan, mode pengalamatan register tidak langsung hanya menggunakan satu referensi memori utama sehingga lebih cepat daripada mode pengalamatan tidak langsung

- Keuntungan:
  Jumlah lokasi memori yang diacu lebih banyak, bergantung pada ukuran register.
- Kerugian:
  Selain mengakses register, diperlukan satu referensi memori.

Register Indirect Addressing

- EA = (R)
- Tempat pengalamatan luas (2n)
- Diagram register indirect addressing



Displacement Addressing

- Menggabungkan kemampuan pengalamatan langsung dan pengalamatan register tidak langsung
- Mode in mensyaratkan instruksi memiliki dua buah field alamat, sedikitnya sebuah field yang eksplisit
- Field eksplisit bernilai A dan field implisit mengarah pada register
- Operand berada pada alamat A ditambahkan isi register

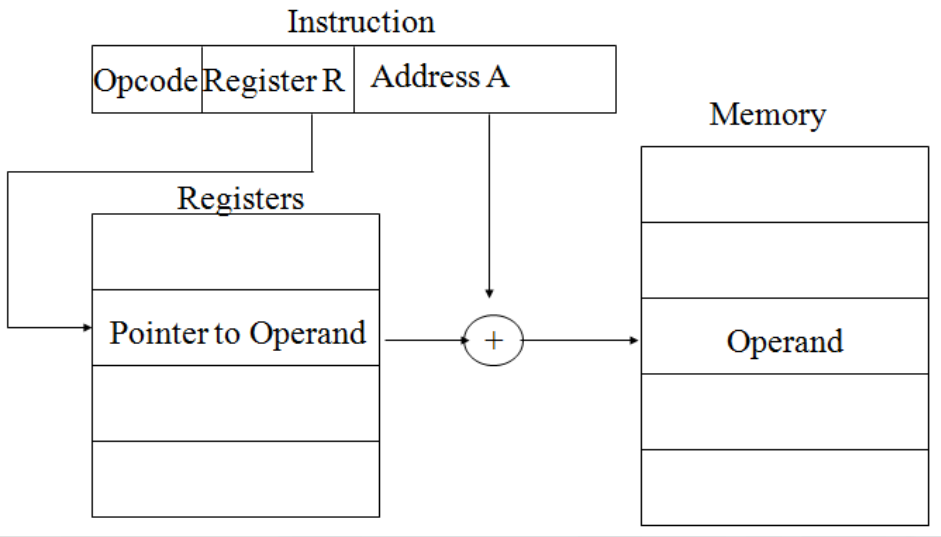
- Keuntungan:
  Lebih fleksibel
- Kerugian:
  Lebih kompleks

Displacement Addressing

- EA = A + (R)

Berdasarkan formula di atas, address field menampung 2 nilai, yaitu:

- A sebagai base value
- R sebagai register yang menampung pertukaran sementara (that holds displacement)
- Atau sebaliknya



Relative Addressing

Register yang direferensi secara implisit adalah progra counter (PC)

- Alamat efektif didapatkan dari alamat instruksi saat itu ditambahkan ke field alamat
- Memanfaatkan konsep lokalitas memori untuk menyediakan operand-operand berikutnya
- Merupakan salah satu versi dari pengalamatan untuk pertukaran (displacement addressing)

R = Program counter, PC
EA = A + (PC)

Keterangan:
Ambil operand dari A dan dari lokasi yang ditunjukkan oleh program counter (PC)

Base-Register Addressing

Mode ini digunakan untuk relokasi program di dalam memori (dari satu area ke area lain). Pada mode pengalamatan pengalamatan base register, instruksi tidak berisi alamat.

Dia memberikan perpindahan relatif terhadap area memori sekarang ke area memori yang lain, base register diisi dengan alamat base baru.

Instruksi tidak perlu dimodifikasi/diubah. Dengan cara ini, keseluruhan program atau suatu segment dari program dapat dipindahkan dari satu area di memori ke yang lain tanpa mempengaruhi instruksi, dengan perubahan sederhana ini base register.

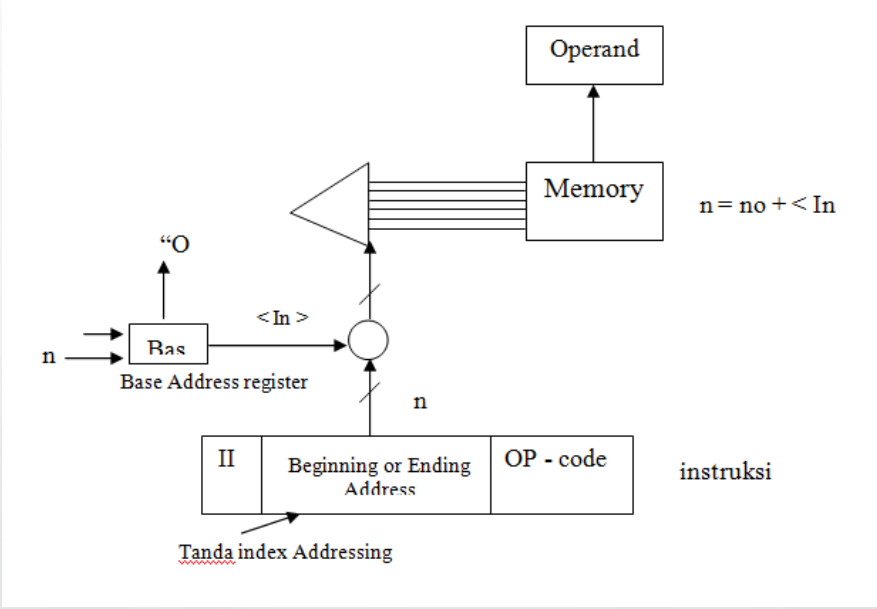
Base register addresing, register yang direferensi berisi sebuah alamat memori, dan field alamat berisi perpindahan dari alamat itu

- Referensi register dapat eksplisit maupun implisit
- Memanfaatkan konsep lokalitas memori

Indexed Addressing

- Indexing dalah field alamat mereferensi alamat memori utama, dan register yang direferensikan berisi pemindahan positif dari alamat tersebut
- Merupakan kebalikan dari mode base register
- Field alamat dianggap sebagai alamat memori dalam indexing
- Manfaat penting dari indexing adalah untuk eksekusi program-program iteratif
- A = dasar
- R = displacement
- EA = A + R
- Baik untuk pengaksesan array
  EA = A + R
  R++

Combinations
- Postindex
  EA = (A) + (R)
- Preindex
  EA = (A+(R))



Stack Addressing

Stack adalah array lokasi yang linear, yang merupakan blok lokasi yang terbalik, sehingga sering disebut juga Last In First Out Queue.

Tidak ada referensi memori di dalam Stack Addressing di mana aplikasi memori yang dimilikinya terbatas.

Stack addressing merupakan bentuk implied addressing. Instruksi-instruksi mesin tidak perlu memiliki referensi memori namun secara implisit beroperasi pada bagian paling atas stack.

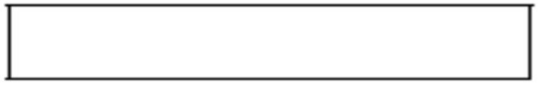
Dua elemen teratas stack dapat berada di dalam register CPU, yang dalam hal ini stack pointer mereferensi ke elemen ketiga stack. Stack pointer tetap berada dalam register.

Operand secara implisit berada pada bagian paling atas stack

Stack Addressing

- Contoh :
- ADD pop dua elemen teratas stack dan jumlahkan

Format instruksi



Operand terletak di puncak stack

- Keuntungan:

Ukuran instruksi kecil

- Kerugian:

Pemakaiannya terbatas

Instruction Format

Format instruksi menentukan layout bit di dalam suatu instruksi yang mencakup opcode, dan (implicit dan explicit) operand.

Pada umumnya lebih dari satu format instruksi di sebuah set instruksi.

Panjang instruksi harus merupakan kelipatan panjang karakter, yang umumnya 8 bit dan kelipatan panjang bilangan fixed point.

Untuk memahami hal ini, kita harus menggunakan istilah word.

Umumnya ukuran word menentukan ukuran bilangan-bilangan fixed point dan berhubungan dengan ukuran transfer memori.

Instruction Format

Panjang instruksi dipengaruhi dan mempengaruhi yaitu : ukuran memori, organisasi memori, struktur bus, kompleksitas CPU, kecepatan CPU.

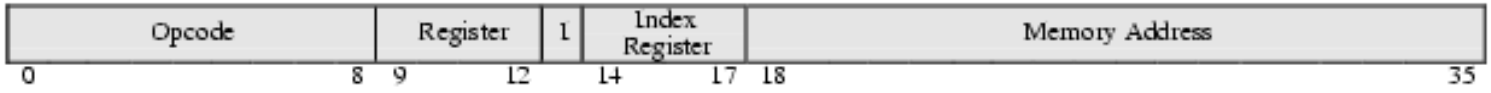
Trade off between powerful instruction repertoire and saving space.

Alokasi bit yang berkaitan dengan penggunaan bit-bit pengalamatan :

1. Jumlah mode pengalamatan dan operand
2. Jumlah set register
3. Register dan Memori
4. Jangkauan Alamat

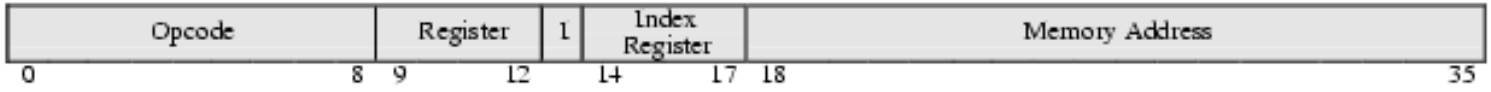
Instruction Format

Format instruksi PDP-10



I = indirect bit

Format instruksi PDP-11



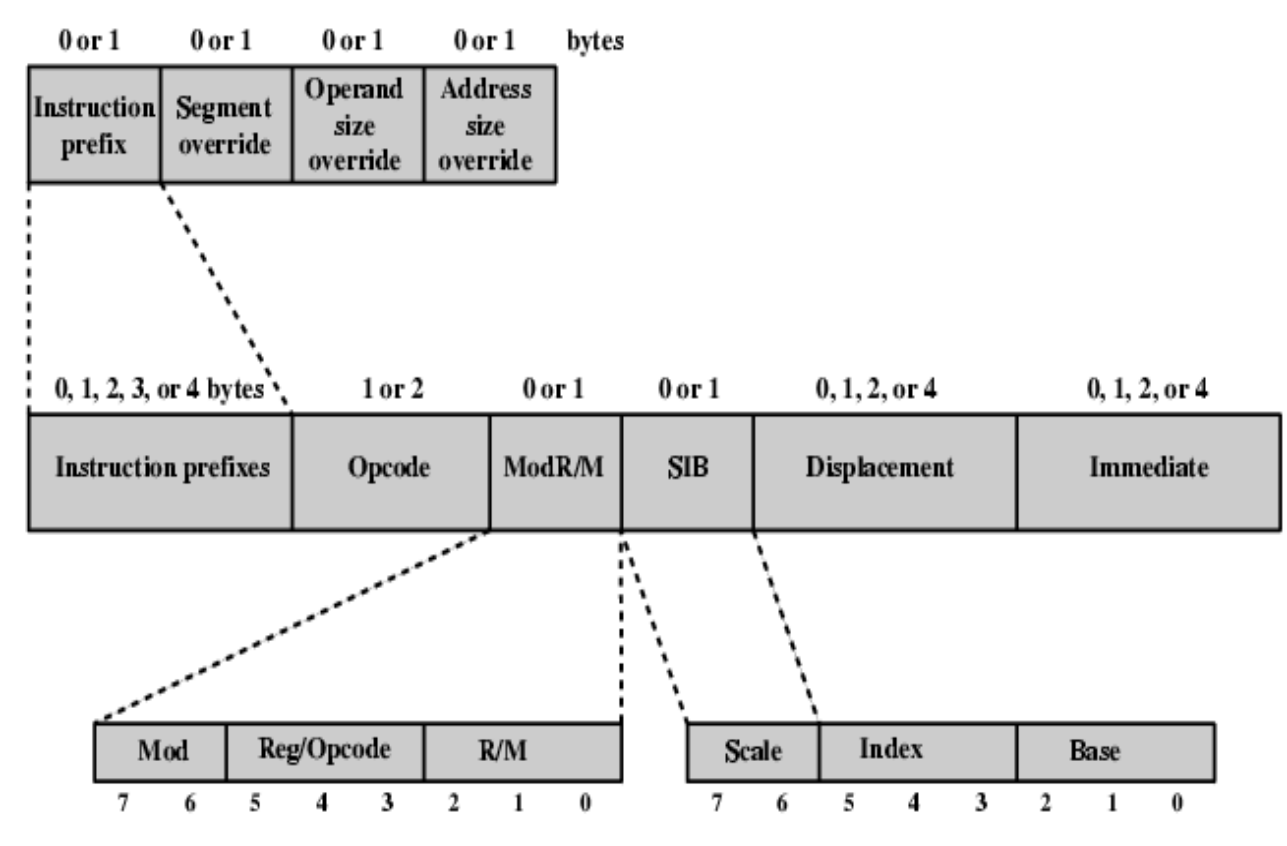
I = indirect bit

Instruction Format

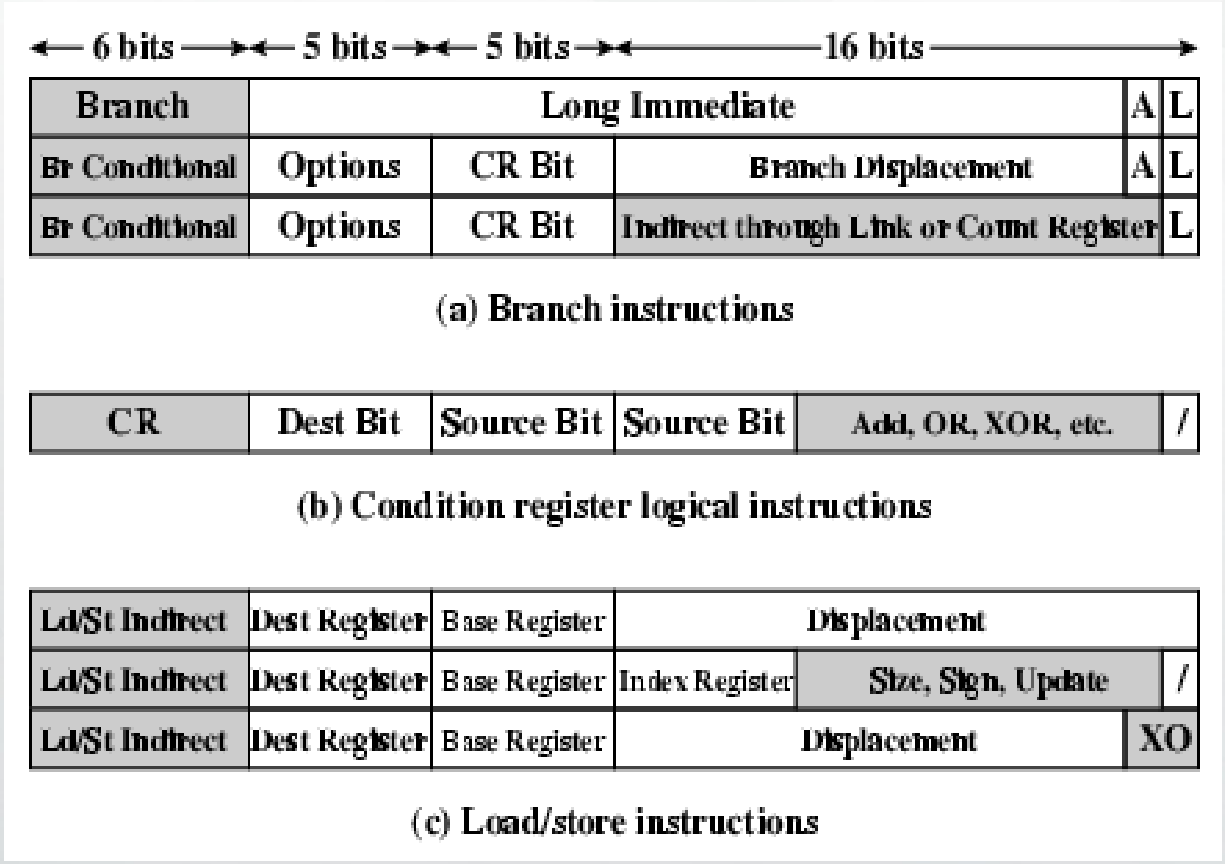
Contoh instruksi VAX

Hexadecimal Format	Explanation	Assembler Notation and Description
8 bits 0 5	Opcode for RSB	RSB Return from subroutine
D 4 5 9	Opcode for CLRL Register R9	CLRL R9 Clear register R9
B 0 C 4 6 4 0 1 A B 1 9	Opcode for MOVW Word displacement mode, Register R4 356 in hexadecimal Byte displacement mode, Register R11 25 in hexadecimal	MOVW 356(R4), 25(R11) Move a word from address that is 356 plus contents of R4 to address that is 25 plus contents of R11
C 1 0 5 5 0 4 2 D F [ ]	Opcode for ADDL3 Short literal 5 Register mode R0 Index prefix R2 Indirect word relative (displacement from PC) Amount of displacement from PC relative to location A	ADDL3 #5, R0, @A[R2] Add 5 to a 32-bit integer in R0 and store the result in location whose address is sum of A and 4 times the contents of R2

Instruction Format : Pentium



Instruction Format : Power PC –1



6 bits — 5 bits — 5 bits — 16 bits

Branch	Long Immediate			A	L
Br Conditional	Options	CR Bit	Branch Displacement	A	L
Br Conditional	Options	CR Bit	Indirect through Link or Count Register		L

(a) Branch instructions

CR	Dest Bit	Source Bit	Source Bit	Add, OR, XOR, etc.	/

(b) Condition register logical instructions

Ld/St Indirect	Dest Register	Base Register	Displacement		
Ld/St Indirect	Dest Register	Base Register	Index Register	Size, Sign, Update	/
Ld/St Indirect	Dest Register	Base Register	Displacement		XO

(c) Load/store instructions

Instruction Format : Power PC –2

Arithmetic	Dest Register	Src Register	Src Register	O	Add, Sub, etc.		R	
Add, Sub, etc.	Dest Register	Src Register	Signed Immediate Value					
Logical	Src Register	Dest Register	Src Register	ADD, OR, XOR, etc.			R	
AND, OR, etc.	Src Register	Dest Register	Unsigned Immediate Value					
Rotate	Src Register	Dest Register	Shift Amt	Mask Begin	Mask End		R	
Rotate or Shift	Src Register	Dest Register	Src Register	Shift Type or Mask			R	
Rotate	Src Register	Dest Register	Shift Amt	Mask	XO	S	R	*
Rotate	Src Register	Dest Register	Src Register	Mask	XO		R	*
Shift	Src Register	Dest Register	Shift Type or Mask			S	R	*

(d) Integer arithmetic, logical, and shift/rotate instructions

Flt sgl/dbl	Dest Register	Src Register	Src Register	Src Register	Fadd, etc.	R

(e) Floating-point arithmetic instructions

A = Absolute or PC relative
L = Link or subroutine
O = Record overflow in XER
R = Record condition in CR1

XO = Opcode extension
S = Part of shift amount field
* = 64-bit implementation only

Terima Kasih

Pustaka : William Stallings, “Computer Organization and Architecture Designing for Performance Eighth Edition”, Prentice Hall, 2019