

ARSITEKTUR DAN ORGANISASI KOMPUTER

ADDRESSING MODE

FTI UNISBANK SEMARANG 2010

Konsep Operasional Processor dan Memori

Suatu karakteristik unik dari memori adalah bahwa memori tersebut harus diorganisasikan dalam suatu *hierarchy*. Pada *hierarchy* tersebut, memori yang berukuran lebih besar dan kecepatannya lebih lambat digunakan untuk mendukung memori yang berukuran kecil tetapi memiliki kecepatan tinggi. *Hierarchy* awal dari memori diawali dari memori yang berukuran kecil, mahal, tetapi kecepatan aksesnya tinggi, dinamakan dengan *cache memory*. Diikuti *hierarchy* berikutnya adalah memori yang berukuran lebih besar, harga lebih murah, tetapi kecepatan aksesnya lebih lambat dari *cache*.

Konsep Operasional Processor dan Memori

Aktivitas dalam komputer diatur oleh instruksi. Untuk melakukan suatu tugas tertentu, suatu program yang berisi daftar instruksi disimpan dalam memori.

Instruksi individu dibawa dari memori ke processor, yang mengeksekusi operasi tertentu. Data yang digunakan sebagai operand juga disimpan dalam memori. Berikut ini adalah contoh dari suatu instruksi :

```
LOAD LOCA, R1  
ADD R1, R0
```

Konsep Operasional Processor dan Memori

LOAD LOCA, R1

ADD R1,R0

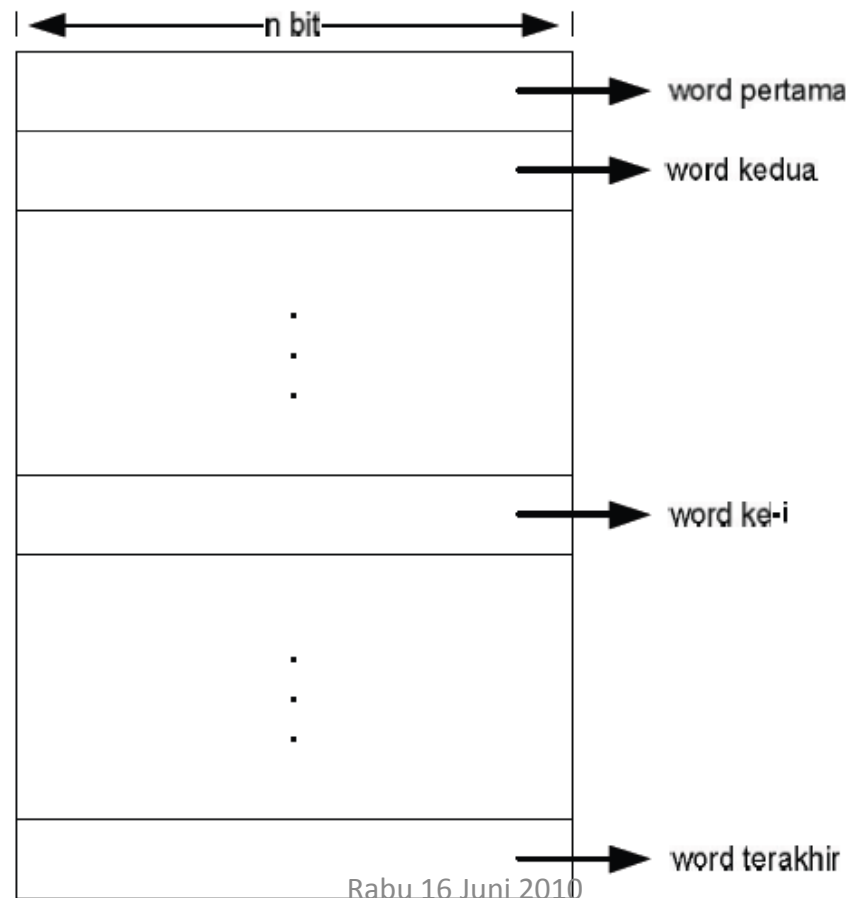
Baris pertama dari instruksi diatas digunakan untuk mentransfer isi lokasi memori *LOCA*, ke register prosessor *R1*, dan instruksi kedua menambahkan isi register *R1* dan *R0* dan menyimpan hasil penjumlahan pada register *R0*. Dua instruksi diatas menghasilkan suatu operasi yang menghancurkan isi register *R1* dan *R0* sebelum instruksi dieksekusi, sedangkan isi memori lokasi *LOCA* tetap dipertahankan.

Lokasi Memori dan Alamat

*Operand bilangan dan karakter, seperti halnya instruksi, disimpan dalam memori komputer. Sekarang kita akan membahas bagaimana memori diatur. Memori terdiri dari jutaan sel penyimpanan, dimana tiap sel tersebut menyimpan suatu bit informasi yang berupa nilai 0 dan 1. Karena bit tunggal mewakili jumlah informasi yang sangat sedikit, maka bit jarang ditangani secara individu. Pendekatan yang umum adalah menanganinya dalam kelompok dengan ukuran tertentu. Untuk tujuan ini, memori tersebut diatur sehingga kelompok n bit disebut word informasi (beberapa referensi menyebut juga dengan nama register), dan n disebut *word length*.*

Lokasi Memori dan Alamat

Memori suatu komputer dapat digambarkan secara skematis sebagai kumpulan word seperti pada gambar berikut.

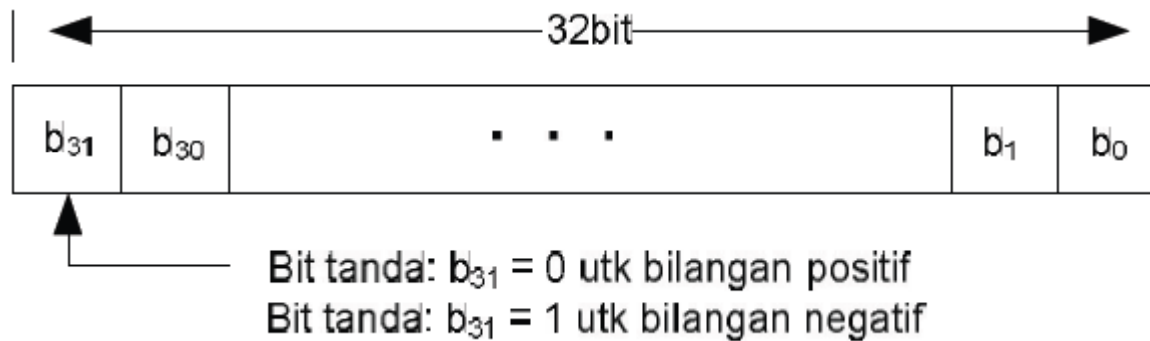


Lokasi Memori dan Alamat

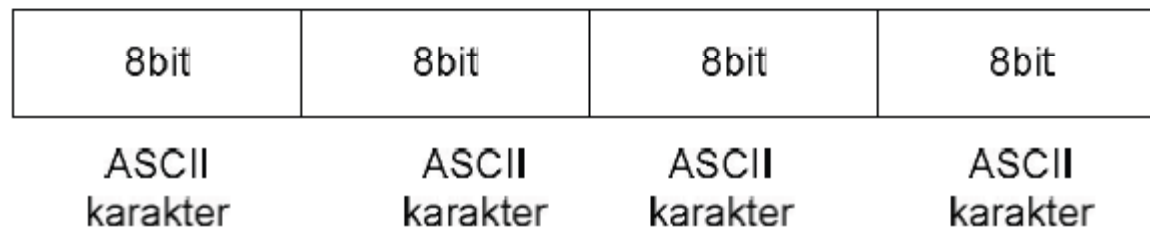
Komputer modern memiliki *word length* yang biasanya berkisar antara 16 – 64 bit. Jika *word length* suatu komputer adalah 32 bit, maka *word tunggal* dapat menyimpan 32bit bilangan komplement atau empat karakter ASCII, masingmasing memiliki 8bit, sebagaimana ditunjukkan pada gambar berikut. Suatu unit 8bit disebut *byte*. *Instruksi mesin* mungkin memerlukan satu atau lebih *word* untuk mewakilinya.

Lokasi Memori dan Alamat

Berikut akan dibahas bagaimana instruksi diencode menjadi word memori pada bagian selanjutnya.



(a) Integer Bertanda



(b) Empat karakter

Lokasi Memori dan Alamat

Mengakses memori untuk menyimpan atau mengambil suatu item informasi, baik berupa word atau byte, memerlukan nama yang berbeda atau alamat tiap lokasi item. Merupakan hal yang biasa menggunakan bilangan dari 0 hingga 2^k-1 , untuk beberapa nilai k yang sesuai, sebagai alamat yang berurutan dalam memori. Alamat 2^k meliputi ruang alamat komputer tersebut, dan memori tersebut dapat memiliki lokasi *addressable hingga 2^k* .

Lokasi Memori dan Alamat

Misalnya alamat 24bit menghasilkan ruang alamat 2^{24} (16.777.216) lokasi. Bilangan ini biasanya ditulis 16M(16Mega), dimana 1M adalah bilangan 2^{20} (1.048.576). Alamat 32bit menghasilkan ruang alamat 2^{32} atau 4G(4Giga) lokasi, dimana 1G adalah 2^{30} . Konvensi yang biasa digunakan adalah K(kilo) untuk bilangan 2^{10} (1024), dan T(tera) untuk bilangan 2^{40} .

Byte Addressability

Sekarang terdapat tiga kuantitas informasi dasar yang telah dijelaskan, yaitu bit, byte, dan word. Byte selalu 8bit, tetapi word length biasanya berada pada rentang 16 hingga 64 bit. Sangat tidak praktis untuk menetapkan alamat yang berbeda bagi lokasi bit individu dalam memori. Penetapan paling praktis adalah dengan alamat yang berurutan mengacu pada lokasi byte yang berurutan dalam memori.

Byte Addressability

Ini merupakan penetapan yang digunakan pada sebagian besar komputer modern. Istilah yang digunakan adalah *byte addressable memory*. Lokasi *byte* memiliki alamat 0,1,2,... sehingga, jika word length suatu mesin adalah 32bit, maka word yang berurutan berada pada alamat 0,4,8, ..., dengan tiap word terdiri dari empat byte.

Penetapan *BigEndian* dan *LittleEndian*

Terdapat dua cara penetapan alamat byte pada word, sebagaimana ditampilkan pada gambar berikut. Nama *bigendian* dipakai jika alamat byte rendah untuk *Most Significant Byte (byte paling kiri)* dari word tersebut. Nama *littleendian* digunakan untuk pengaturan sebaliknya, yaitu alamat byte rendah dipakai untuk *less significant byte (byte paling kanan)* dari word tersebut.

Penetapan *BigEndian* dan *LittleEndian*

Kata “*most significant*” dan “*less significant*” digunakan dalam kaitannya dengan *weight* (pangkat 2) yang ditetapkan pada bit pada saat word tersebut menyatakan suatu bilangan. Penetapan *BigEndian* dan *LittleEndian* digunakan dalam mesin komersial. Pada kedua kasus tersebut, alamat byte 0,4,8,... digunakan sebagai alamat word yang berurutan dalam memori dan merupakan alamat yang digunakan pada saat menetapkan operasi baca tulis memori untuk word.

Penetapan *BigEndian* dan *LittleEndian*

Selain menentukan urutan alamat byte dalam word, juga perlu menentukan label bit atau word. Konvensi yang paling umum, ditampilkan pada gambar diatas. Konvensi tersebut merupakan penyusunan paling alami untuk data numerik. Penyusunan yang sama juga digunakan untuk menetapkan label bit dalam byte, yaitu b7,b6,...,b0, dari kiri ke kanan. Namun ada pula komputer yang menggunakan penyusunan sebaliknya.

Penetapan *BigEndian* dan *LittleEndian*

Alamat word	Alamat byte			
0	0	1	2	3
4	4	5	6	7
	⋮			
$2^k - 4$	$2^k - 4$	$2^k - 3$	$2^k - 2$	$2^k - 1$

Big-Endian

Alamat word	Alamat byte			
0	3	2	1	0
4	7	6	5	4
	⋮			
$2^k - 4$	$2^k - 1$	$2^k - 2$	$2^k - 3$	$2^k - 4$

Little-Endian

Mengakses Bilangan, Karakter, dan String Karakter

Sebuah bilangan biasanya memiliki satu word. Bilangan tersebut dapat diakses dalam memori menetapkan alamat wordnya. Seperti halnya karakter individu dapat diakses melalui alamat bytenya. Pada banyak aplikasi, diperlukan penanganan string karakter variable length. Awal string diindikasikan dengan menyatakan byte yang berisi karakter pertama pada alamat tersebut.

Mode Pengalamatan

Seluruh Informasi yang diperlukan oleh operasi apapun yang dilakukan oleh CPU harus dialamati. Dalam Ilmu Komputer, informasi tersebut dinamakan *operand*. Seluruh operasi yang dipakai pada prosesor sedikitnya memiliki 2 tipe informasi. Instruksi yang dipakai, diencode dan dinamakan *opcode*, dan informasi alamat diencode dan dinamakan *address*.

Mode Pengalamatan

Instruksi dapat diklasifikasikan berdasarkan jumlah *operand* yaitu : *threeaddress* (tiga alamat), *two address* (dua alamat), *oneandhalfaddress* (satusetengah alamat), *one address* (satu alamat), *zero address* (nol alamat). Pada contoh-contoh berikut, instruksi yang digunakan menggunakan format *operasi, sumber, tujuan* untuk mewakili instruksi apapun.

Mode Pengalamatan

'Operasi' mewakili operasi-operasi yang digunakan, contohnya adalah add, subtract, write, atau read.
'Sumber' mewakili operand sumber. Operand sumber dapat berupa konstanta, nilai yang disimpan pada register, atau nilai yang disimpan pada memori.
'Tujuan' mewakili tempat dimana hasil operasi disimpan, bisa di register atau di memori.

Mode Pengalamatan

Instruksi tiga alamat memiliki bentuk *operasi add1, add2, add3*. Pada bentuk ini, tiap *add1, add2, add3* menunjuk ke register atau ke memori tertentu.

Sebagai contoh, instruksi *ADD R1,R2,R3*. Instruksi ini mengindikasikan bahwa operasi yang dilakukan adalah *Addition (Penjumlahan)*. Instruksi ini juga mengindikasikan bahwa data yang dijumlahkan adalah data yang tersimpan di register *R1 dan R2*, dan hasil penjumlahan disimpan pada register *R3*.

Mode Pengalamatan

Contoh dari Instruksi tiga alamat yang menggunakan lokasi memori berbentuk *ADD A,B,C*. Instruksi tersebut menambahkan data yang ada pada memori lokasi *A* dan *B*, dan menyimpan hasilnya pada memori lokasi *C*.

Mode Pengalamatan

Instruksi dua alamat memiliki bentuk *operasi add1, add2*. Pada bentuk ini, tiap *add1, add2*, menunjuk ke register atau ke memori tertentu. Sebagai contoh, instruksi *ADD R1,R2*. Instruksi ini mengindikasikan bahwa data yang dijumlahkan adalah data yang tersimpan di register *R1* dan *R2*, dan hasil penjumlahan disimpan pada register *R2*. Contoh dari Instruksi dua alamat yang menggunakan lokasi memori berbentuk *ADD A,B*. Instruksi tersebut menambahkan data yang ada pada memori lokasi *A* dan *B*, dan menyimpan hasilnya pada memori lokasi *B*.

Mode Pengalamatan

Instruksi satu alamat memiliki bentuk *ADD R1*. Pada kasus berikut, instruksinya menunjuk ke register, dinamakan *Accumulator Racc*. Data dari *Accumulator* ditambahkan dengan data dari register *R1*, kemudian hasilnya disimpan pada *Accumulator*. Jika data di memori yang digunakan, maka bentuk instruksinya adalah *ADD B*. Pada bentuk ini, operasinya digunakan untuk menjumlahkan data dari *Accumulator* dengan data dari memori lokasi *B*, kemudian hasil penjumlahan disimpan pada *Accumulator*. Instruksi *ADD R1* ekuivalen dengan instruksi tiga alamat *ADD R1,Racc,Racc* atau instruksi dua alamat *ADD R1,Racc*.

Mode Pengalamatan

Diantara instruksi dua dan satu alamat, terdapat instruksi satusetengah alamat. *ADD B,R1* adalah termasuk instruksi satusetengah alamat. Pada bentuk ini, operasinya adalah menambah data Register *R1* dengan data dari memori lokasi *B*, kemudian menyimpan hasil penjumlahan di register *R1*. *Faktanya, instruksi tersebut* menggunakan dua tipe pengalamatan, yaitu register dan lokasi memori, hal ini disebut dengan instruksi satusetengah alamat, karena pengalamatan register membutuhkan jumlah bit yang lebih sedikit dibandingkan dengan bit yang dibutuhkan untuk pengalamatan memori.

Mode Pengalamatan

Klasifikasi instruksi	Contoh
Tiga alamat	<i>ADD R1,R2,R3</i> <i>ADD A,B,C</i>
Dua alamat	<i>ADD R1,R2</i> <i>ADD A,B</i>
Satu-setengah alamat	<i>ADD B,R1</i>
Satu alamat	<i>ADD R1</i>

TUGAS

Buat tulisan / artikel yg menjelaskan tentang :

1. *Immediate Addressing*
2. *Direct (Absolute) Addressing*
3. *Indirect Addressing*
4. *Indexed Addressing*

Print pada kertas dan dikumpulkan paling lambat hari Selasa / 22 Juni 2010 jam 12.00