

BAHASA ASSEMBLY

ARSITEKTUR DAN ORGANISASI
KOMPUTER

FTI UNISBANK 2010

BAHASA ASSEMBLY

- Instruksi mesin dinyatakan dengan pola 0 dan 1. Pola semacam itu sangat sulit untuk dijelaskan pada saat membahas atau menyiapkan program. Oleh karena itu, kita menggunakan nama simbolik untuk menyatakan pola tersebut. Sejauh ini kita telah menggunakan katakata biasa seperti Move, Add, Increment, dan Branch, untuk instruksi operasi yang menyatakan pola kode biner yang sesuai.

ASSEMBLY

Pada saat menulis program untuk komputer tertentu, kata kata tersebut biasanya diganti dengan akronim yang disebut mnemonic, seperti MOV, ADD, INC, dan BR Serupa dengan kita menggunakan notasi R3 untuk mengacu pada register 3, dan LOC untuk mengacu pada lokasi memori. Nama-nama simbolik dan aturan penggunaannya yg membentuk bahasa pemrograman, biasanya disebut sebagai bahasa assembly. Sedangkan aturan untuk menggunakan mnemonic dalam spesifikasi instruksi dan program lengkap disebut syntax bahasa.

ASSEMBLER

Program yang ditulis dalam bahasa assembly dapat secara otomatis ditranslasikan ke rangkaian instruksi mesin oleh suatu program yang disebut assembler. Program assembler adalah salah satu kumpulan program yang merupakan bagian dari software sistem. Assembler, seperti halnya program yang lain, disimpan sebagai rangkaian instruksi mesin dalam memori komputer

ASSEMBLER

Program user biasanya dimasukkan ke dalam komputer melalui keyboard dan disimpan dalam memori atau disk magnetik. Pada titik ini, program user hanyalah kumpulan baris karakter alfanumerik. Pada saat program assembler dieksekusi, program tersebut membaca program user, menganalisisnya, dan kemudian menghasilkan program bahasa mesin yang diinginkan. Bahasa mesin tersebut berisi pola 0 dan 1 yang menetapkan instruksi yang akan dieksekusi oleh komputer tersebut. Program user dalam format teks alfanumerik aslinya disebut source program, dan program bahasa mesin yang diassemble disebut object program.

ASSEMBLER

Bahasa assembly untuk suatu komputer mungkin case sensitive atau mungkin tidak, sehingga, komputer tersebut bisa membedakan antara huruf kapital dan huruf kecil atau tidak dapat. Kita akan menggunakan huruf kapital utk menunjukkan semua nama dan label dalam contoh untuk dapat meningkatkan kemudahan pembacaan teks. Misalnya, kita akan menuliskan instruksi Move sebagai berikut

```
MOVE R0, SUM
```

ASSEMBLER

MOVE R0, SUM

MOVE mnemonic menyatakan pola biner, atau opcode, untuk operasi yang dilakukan oleh instruksi tersebut. Assembler mentranslasi mnemonic ini menjadi Opcode biner yang dipahami komputer. Mnemonic Opcode diikuti oleh setidaknya satu karakter spasi kosong.

Kemudian informasi yang menyatakan operand ditetapkan. Dalam contoh diatas, source operand berada dalam register R0. Informasi ini diikuti oleh spesifikasi destination operand, dipisah dari source operand dengan koma, tanpa jeda kosong. Destination operand berada dalam lokasi memori yang alamat binernya dinyatakan dengan nama SUM.

ASSEMBLER

ADD #5, R3

menambahkan bilangan 5 ke isi register R3 dan meletakkan hasilnya kembali ke register R3. tanda sharp bukanlah cara satu satunya untuk menunjukkan mode pengalamatan Immediate. Dalam beberapa bahasa assembly, mode pengalamatan yang dimaksud dinyatakan dalam mnemonic OPcode. Dalam hal ini, suatu instruksi memiliki mnemonic Opcode yang berbeda untuk mode pengalamatan yang berbeda.

ASSEMBLER

Pengalamatan Indirect biasanya dinyatakan dengan meletakkan tanda kurung di sekitar nama atau simbol yang menunjukkan pointer ke operand. Misalnya, jika angka 5 ditempatkan dalam lokasi memori yang alamatnya disimpan dalam register R2, maka aksi yang diinginkan dapat ditetapkan sebagai berikut

```
MOVE #5, (R2)
```

atau

```
MOVE 5, (R2)
```

ASSEMBLER DIRECTIVE

Selain menyediakan mekanisme untuk menyatakan instruksi dalam suatu program, bahasa assembly memungkinkan programmer untuk menetapkan informasi lain yang diperlukan untuk mentranslasikan source program ke dalam object program.

Kita telah menyebutkan bahwa kita perlu untuk menetapkan nilai numerik ke tiap nama yang digunakan dalam program. Misalkan nama SUM digunakan untuk menyatakan nilai 200. Fakta ini mungkin disampaikan ke program assembler melalui pernyataan seperti

```
SUM EQU 200
```

ASSEMBLER DIRECTIVE

SUM EQU 200

Pernyataan ini tidak menunjuk suatu instruksi yang akan dieksekusi pada saat object program dijalankan; sebenarnya, bahkan tidak akan muncul dalam object program. Hanya memberitahu assembler bahwa nama SUM seharusnya digantikan dengan nilai 200 kapanpun muncul dalam program. Pernyataan semacam itu, yang disebut assembler directive (atau perintah), digunakan oleh assembler pada saat mentranslasikan source program menjadi object program.

ASSEMBLER DIRECTIVE

	100	Move	N, R1
	104	Move	#NUM1, R2
	108	Clear	R0
LOOP	112	Add	(R2), R0
	116	Add	#4, R2
	120	Decrement	R1
	124	Branch > 0	LOOP
	128	Move	R0, SUM
	132		
			.
			.
			.
SUM	200		
N	204	100	
NUM1	208		
NUM2	212		
			.

ASSEMBLER DIRECTIVE

Untuk menjalankan program pada komputer, maka kita perlu untuk menuliskan source codenya dalam bahasa assembly yang diperlukan, menetapkan semua informasi yang diperlukan untuk menghasilkan object program yang sesuai. Misalkan tiap instruksi dan tiap item data memiliki satu word memori. Juga asumsikan bahwa memori tersebut adalah byte addressable dan word lengthnya adalah 32 bit. Misalkan juga bahwa object program diload dalam memori utama sebagaimana yang ditunjukkan dalam gambar. Gambar diatas menunjukkan alamat memori dimana instruksi mesin dan item data yang diminta didapatkan setelah program diload untuk eksekusi.

ASSEMBLER DIRECTIVE

Jika assembler adalah untuk menghasilkan object program sesuai dengan pengaturan ini, maka harus mengetahui

- Bagaimana menginterpretasikan nama tersebut
- Dimana harus menempatkan instruksi tersebut dalam memori
- Dimana harus menempatkan operand data dalam memori

ASSEMBLER DIRECTIVE

	Label alamat memori	Operasi	Pengalamatan atau informasi data
Assembler directives	sum	EQU	200
		ORIGIN	204
	N	DATAWORD	100
	NUM1	RESERVE	400
Statements that generate machine instructions		ORIGIN	100
	START	MOVE	N,R1
		MOVE	#NUM1,R2
		CLR	RO
	LOOP	ADD	(R2),RO
		ADD	q4,R2
		DEC	R1
	BGTZ	LOOP	

ASSEMBLER DIRECTIVE

Kebanyakan bahasa assembly meminta pernyataan dalam source program dituliskan dalam bentuk

Label Operation Operand Comment

ASSEMBLER DIRECTIVE

Untuk menjalankan program pada komputer, maka kita perlu untuk menuliskan source codenya dalam bahasa assembly yang diperlukan, menetapkan semua informasi yang diperlukan untuk menghasilkan object program yang sesuai. Misalkan tiap instruksi dan tiap item data memiliki satu word memori. Juga asumsikan bahwa memori tersebut adalah byte addressable dan word lengthnya adalah 32 bit. Misalkan juga bahwa object program diload dalam memori utama sebagaimana yang ditunjukkan dalam gambar. Gambar diatas menunjukkan alamat memori dimana instruksi mesin dan item data yang diminta didapatkan setelah program diload untuk eksekusi.

ASSEMBLER DIRECTIVE

Empat field tersebut dipisahkan oleh delimiter yang sesuai, biasanya satu atau lebih karakter kosong. Label adalah nama opsional yang dihubungkan dengan alamat memori dimana instruksi bahasa mesin yang dihasilkan dari pernyataan tersebut akan diload.

Label juga dapat dihubungkan dengan alamat item data. Pada Gambar diatas ada beberapa contoh label yg dipakai. Contoh label adalah:

SUM, N, NUM1, START, dan LOOP. dll

ASSEMBLER DIRECTIVE

Field Operation berisi mnemonic Opcode dari instruksi yang dimaksud atau directive assembler. Field Operand berisi informasi pengalamatan untuk mengakses satu atau dua operand, tergantung pada tipe informasinya. Field Comment diabaikan oleh program assembly. Field tersebut digunakan untuk tujuan dokumentasi sehingga program lebih mudah dipahami.

ASSEMBLY DAN EKSEKUSI PROGRAM

Source program yang ditulis dalam bahasa assembly harus diassemble menjadi object program bahasa mesin sebelum dapat dieksekusi. Hal ini dilakukan oleh program assembler, yang mengganti semua simbol untuk mode operasi dan pengalamatan dengan kode biner yang digunakan dalam instruksi mesin, dan mengganti semua nama dan label dengan nilai sebenarnya.

NOTASI BILANGAN

Pada saat berhadapan dengan nilai numerik, seringkali lebih mudah untuk menggunakan notasi desimal yang telah dikenal. Tentu saja, nilai tersebut disimpan dalam komputer sebagai bilangan biner. Pada beberapa situasi, lebih mudah untuk menetapkan pola biner secara langsung.

Kebanyakan assembler memungkinkan bilangan numerik dinyatakan dengan berbagai cara yang berbeda, menggunakan konvensi yang ditetapkan oleh syntax bahasa assembly. Misalkan, bilangan 93, yang dinyatakan dengan bilangan biner 8bit 01011101.

NOTASI BILANGAN

Jika nilai ini digunakan sebagai operand Immediate, maka dapat dinyatakan sebagai bilangan desimal, sebagaimana dalam instruksi

ADD #93, R1

atau sebagai bilangan biner yang diidentifikasi dengan simbol awalan seperti tanda persen, seperti

ADD #%01011101, R1

NOTASI BILANGAN

Bilangan biner dapat dituliskan lebih padat sebagai bilangan heksadesimal, atau hex, dengan empat bit dinyatakan dengan digit hex tunggal. Notasi hex adalah ekstensi langsung dari kode BCD yang terdapat dalam Apendiks E. Sepuluh pola pertama 0000, 0001, ..., 1001, dinyatakan dengan digit 0, 1, ..., 9 sebagaimana dalam BCD. Sisa enam pola 4bit, 1010, 1011, ..., 1111, dinyatakan dengan huruf A, B, ..., F.

Dalam representasi heksadesimal, nilai desimal 93 menjadi 5D. Dalam bahasa assembly, representasi hex seringkali diidentifikasi dengan awalan tanda dolar. Sehingga kita menuliskannya

```
ADD #$5D, R1
```