Intermediate Code Form

Kriteria Pemilihan Intermediate Code Form (ICM) yang tepat :

- Kemudahan dalam membangun ICM sehingga memudahkan proses analisis dan interpretasi pada pass II
- Storage area yang ekonomis, karena mereduksi kebutuhan storage pada assembler sehingga lebih padat

Intermediate Code Form (cont.)

Contoh :

Stmt				Address	Instruction		
1		START	200			AD #5	200,
2		LOAD	= '5'	200	+ 04 211	04	L#1
3		STORE	A	201	+ 05 217	05	S#1
4	LOOP	LOAD	A	202	+ 04 217	04	S#1
5		SUB	= '1'	203	+ 02 212	02	L#2
6		DS	1	204	+ 05 218	DL#1	1,
		-		-			
12		TRANS	NEXT	210	+ 06 213	06	S#3
13		LTORG				AD#3	

Assembler program | Machine program | Intermediate Code

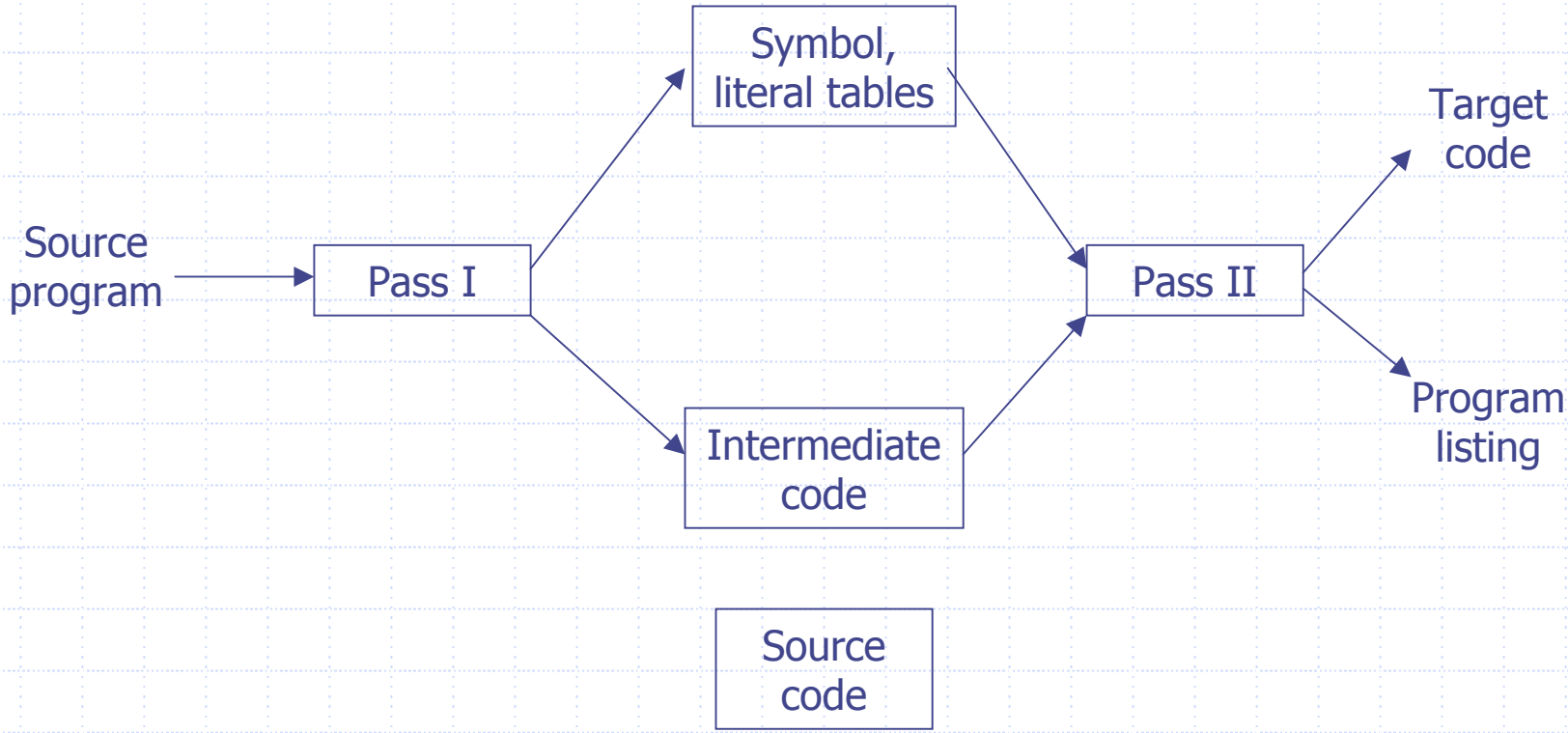
Listing dan Indikasi Kesalahan

- Indikasi kesalahan berpengaruh pada **efektivitas**, **kebutuhan storage** dan **kecepatan assembly**
- Jika listing dan indikasi kesalahan dijalankan pada pass I : begitu prosess pada source statement selesai dikerjakan langsung dapat dicetak untuk diketahui posisi dari kesalahan tersebut.
- Jika listing dan indikasi kesalahan dijalankan pada pass II : source program disimpan dalam secondary storage device dalam bentuk file.

Pass II Assembler

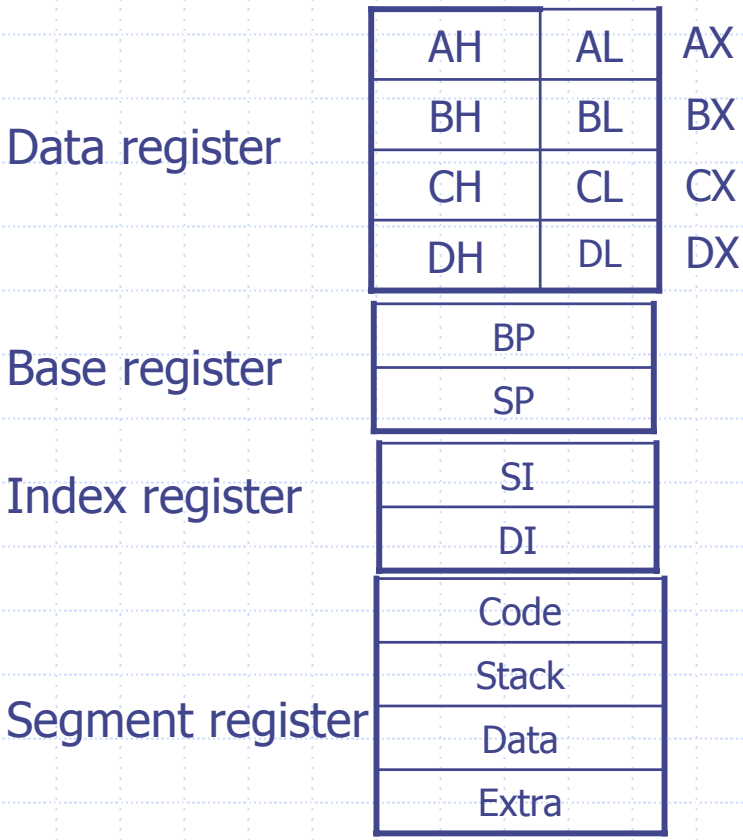
- Dua hal dasar yang dilakukan pada pass II :
  - Men-'generate machine languange program
  - Melakukan penyesuaian bentuk agar sesuai dengan kebutuhan linkage editor atau loader
  - Output yang dihasilkan adalah object module, yang terdiri dari 2 komponen :
    - Target program
    - Informasi tambahan yang relevan untuk linkage editor atau loader

Skematik "Two Pass Assembler"



Single Pass Assembler untuk IBM PC (Intel 8088)

- Arsitektur Intel 8088



Single Pass Assembler untuk IBM PC (Intel 8088) (cont.)

- Addressing Mode 8088

Addressing Mode	Example	Remarks
Immediate	MOVE SUM, 1234	Data = 1234 H
Register	MOVE SUM, AX	AX contains the data
Direct	MOV SUM, [1234H]	Data displacement = 1234H
Register direct	MOV SUM, [BX] MOV SUM, CS: [BX]	Data displacement = (BX) Segment base = (CS), Data displacement = (BX)
Based	MOV SUM, 12H[BX]	Data displacement =12H+(BX)
Indexed	MOV SUM, 34H[SI]	Data displacement =34H+(SI)
Based and indexed	MOV SUM, 56[SI][BX]	Data displacement=56H+(SI)+(BX)

Single Pass Assembler untuk IBM PC (Intel 8088) (cont.)

Intel 8088 Instruction

1. Arithemtic Instruction

a) Register/Memory to Register

b) Immediate to Register/Memory

c) Immediate to Accumulator

Single Pass Assembler untuk IBM PC (Intel 8088) (cont.)

Contoh Instruksi

St.No.	Assembly Statement	Opcode	d	w	Mod	reg	r/m	Data/ displacement
1	ADD AL, BL	000000	0	0	11	011	000	
2	ADD AL, 12H(SI)	000000	1	0	01	000	100	00010010
3	ADD AX, 3456H	100000	0	1	11	000	000	01010110 00110100
4	ADD AX, 3456H	000000	0	1	01010110			00110100

Single Pass Assembler untuk IBM PC (Intel 8088) (cont.)

2. Control Transfer Instruction
   a) Call, Jumps dan Return
   b) Iteration Control Instruction

Format :

a) Intra-segment

Opcode	Disp.low	Disp.high

b) Inter-segment

Opcode	offset	offset	segment	base

c) Indirect

Opcode	mod 100 r/m	disp.low	disp.high

Contoh :

```
MOVSI, 100H          ; Load SI with source address
MOV DI, 200H         ; Load DI with destination address
MOV CX, 50H          ; Load Count (= no.of bytes)
CLD                  ; Clear direction flag
REP MOVSB            ; Move 80 bytes
```

Single Pass Assembler untuk IBM PC (Intel 8088) (cont.)

- Assembly Languange pada Intel 8088

a) Format Statement

Label : opcode operand(s); comment string

b) Assembler Directive

- ORIGIN, EQU, END
- EQU, PURGE EQU
- SEGMENT, ENDS, ASSUME
- PROC, ENDP, NEAR, FAR
- PUBLIC, EXTRN

c) Analytic Operators

- SEG, OFFSET, TYPE, SIZE, LENGTH

d) Synthetic Operators

- PTR, THIS

Macros dan Macro Processor

Macro Assembler :

Fasilitas yang disediakan oleh assembly languange untuk memperluas jangkauan operasinya.

- Assembly languange programming biasanya tersusun atas urutan instruksi yang menempati sejumlah tempat pada program
- Urutan yang umum digunakan pada proses aritmetika yang dilakukan oleh accumulator adalah :
  1) Ambil data (Load) dari accumulator
  2) Tambahkan (increment) data tersebut dengan suatu constanta atau nilai dari variable
  3) Simpan (store) nilai baru yang dihasilkan pada lokasi penyimpanan yang telah ditentukan

LOAD – ADD – STORE

- Urutan perintah LOAD-ADD-STORE dapat disederhanakan dengan perintah **INCR**

Macros dan Macro Processor (cont.)

Secara umum penulisan Macro Assembler terdiri dari :

1. **Macro Definition**
2. **Macro Expansion**

Contoh :

```
MACRO
INCR &X, &Y
LOAD    &X
ADD     &Y
STORE   &Y
MEND
-
-
INCR    A, B
-
-
END
```

Macro Definition

```
LOAD    A
ADD     B
STORE   A
```

Macro Expansion

Macro Definition

- Macro definition ditempatkan diantara perintah MACRO dan MEND
- Setiap unit baru suatu operasi atau urutan sequence assembly languange harus terlebih dahulu didefinisikan di area Macro Definition
- Macro Definition Unit

```
MACRO                    Macro header statement
INCR &X, &Y              Macro prototype statement
LOAD    &X
ADD     &Y               Model statement
STORE   &Y
MEND                     End of definition unit
```

Macro Definition (cont.)

- **Macro header statement**, mengindikasikan penggunaan macro definition unit
- **Macro prototype statement**, mengindikasikan bagaimana operand dalam macro call ditulis
  - ✓ Operand di sini disebut **parameter** atau argument
  - ✓ Penulisan parameter yang dimulai dengan simbol '&' disebut **formal parameter**
  - ✓ Penulisan parameter yang tidak dimulai dengan simbol '&' disebut **actual parameter**
  - ✓ Actual parameter selalu dipasangkan dengan formal parameter

  Contoh :

  ```
  INCR    &X, &Y      ....prototype
  INCR    A, B        ...macro call
  ```
- Model statement, assembly statement yang mengganti macro call sebagai hasil macro expansion

Macro Definition (cont.)

Keyword Parameter

```
        MACRO
        CALC        &X =,&Y =, &OP = MULT, &LAB =
&LAB    LOAD  &X
        &OP         &Y
        STORE       &C
        MEND
        -
        -
        CALC        Y = B, LAN = LOOP, X = A
```

```
+ LOOP LOAD A
+       MULT B
+       STORE A
```

Macro Definition (cont.)

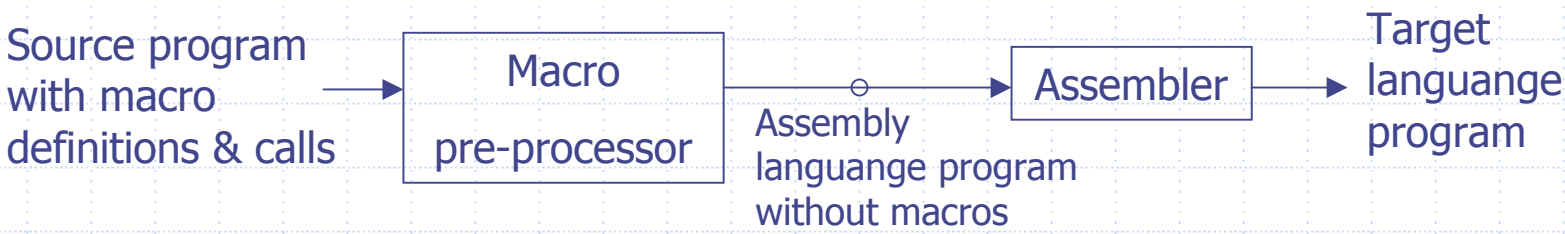
Conditional Assembly

```
        MACRO
        EVAL    &X, &Y, &Z
        AIF     (&Y EQ&X) . ONLY
        LOAD    &X
        SUB     &Y
        ADD     &Z
        AGO     .OVER
.ONLY   LOAD    &Z
.OVER   MEND
```

Macro Expansion

- Dalam macro expansion dilakukan penggantian statement demi statement yang berada dalam satu urutan ke dalam macro

Skematik Macro Expansion



Desain Macro Pre-Processor

- Step-1 : melakukan scanning terhadap semua macro definiton satu persatu :
  i. Memasukan setiap nama ke dalam Macro Name Table (MNT)
  ii. Menyimpan macro definition ke dalam Macro Definition Table (MDT)
  iii. Memberi informasi tambahan ke dalam MNT yang dapat membantu menemukan MDT
- Step-2 : menguji semua statement source program assembly untuk mendeteksimacro call :
  i. Mengalokasikan macro ke dalam MNT
  ii. Memperoleh informasi dari MNT berkaitan dengan posisi macro definition di MDT
  iii. Memproses macro call statement untuk menentukan korespondensi antara semua format parameter dan nilainya (actual parameter)
  iv. Mengembangkan macro call dengan mengikuti prosedur pada step-3
- Step-3 : memproses statement dalam macro definition sesuai yang ditemukan pada MDT pada expansion time order hingga statement MEND

Desain Macro Assembler

Pass Structure Macro Assembler

- Pass 1
  - ✓ Pemrosesan macro definition : membangun MNT, MDT dan KPT
  - ✓ Mengumpulkan simbol dibutuhkan dan informasi terkait
- Pass 2 :
  - ✓ Macro expansion
  - ✓ Location Counter Processing dan melengkapi informasi pada SYMTAB
  - ✓ Memproses litteral
  - ✓ Intermediate Code Generation
- Pass 3 :
  - ✓ Men'generate target code