



6. 기본 컴퓨터 프로그래밍

목표

- 기본 컴퓨터의 프로그래밍 환경에 대해 이해한다.



기계어, MACHINE LANGUAGE



기본 컴퓨터를 위한 기계어 프로그램

Location	Instruction code			
0	0010	0000	0000	0100
1	0001	0000	0000	0101
10	0011	0000	0000	0110
11	0111	0000	0000	0001
100	0000	0000	0101	0011
101	1111	1111	1110	1001
110	0000	0000	0000	0000

무엇을 하는 프로그램인가?



프로그래밍 언어

- 프로그램 작성을 위해 사용하는 언어
- 종류
 - 기계어, Machine Language
 - 어셈블리어, Assembly Language
 - 고급 프로그래밍 언어, High Level Language, HLL



기계어

- 메모리에 실제로 저장되는 형태로 명령어를 표현
 - 이진 형식
- 컴퓨터가 실제로 실행할 수 있는 명령어

Location	Instruction code			
0	0010	0000	0000	0100
1	0001	0000	0000	0101
10	0011	0000	0000	0110
11	0111	0000	0000	0001
100	0000	0000	0101	0011
101	1111	1111	1110	1001
110	0000	0000	0000	0000



어셈블리어

- 명령어의 Op-코드 필드, 주소 필드 등을 기호로 사용하여 표현
- 컴퓨터가 실행할 수 있는 형식이 아님
→ 기계어로 변환하는 프로그램이 필요
- 어셈블러, Assembler
 - 어셈블리어 프로그램을 기계어 프로그램으로 변환



어셈블리어

Location	Instruction		Comments
000	LDA	004	Load first operand into AC
001	ADD	005	Add second operand to AC
002	STA	006	Store sum in location 006
003	HLT		Halt computer
004	0053		First operand
005	FFE9		Second operand(negative)
006	0000		Store sum here

→ 주소 등을 여전히 프로그래머가 관리해야 함.



기본 컴퓨터를 위한 어셈블리 언어

언어 규칙

- 어셈블리어 명령문으로 프로그램 작성
- 어셈블리어 명령문

LABEL, 명령문 / 주석문

LABEL은 주소를 직접 지정하지
않고 3글자 이하의 기호(symbol)로
표시함.
옵션.

주석문은 /로 시작을 함.
주석문을 옵션.

명령어(instruction) 또는 의사 명령어(Pseudo code)



의사 명령어, Pseudo Code

- 어셈블러가 어셈블리어 프로그램을 기계어로 번역할 때 필요한 정보 제공을 목적으로 함.
- 기계어 명령어로 번역되지 않음
- 의사 명령어 종류

Symbol	Information for the Assembler
ORG N	Hexadecimal number N is the memory location for the instruction or operand listed in the following line
END	Denotes the end of symbolic program
DEC N	Signed decimal number N to be converted to binary
HEX N	Hexadecimal number N to be converted to binary

어셈블리어 프로그램의 예 :

Location	Instruction	Comment
000	LDA 004	ORG 0 /Origin of program is location 0
001	ADD 005	LDA A /Load operand from location A
002	STA 006	ADD B /Add operand from location B
003	HLT	STA C /Store sum in location C
004	0053	HLT /Halt computer
005	FFE9	A, DEC 83 /Decimal operand
006	0000	B, DEC -23 /Decimal operand
		C, DEC 0 /Sum stored in location C
		END /End of symbolic program

어셈블러의 역할

```
ORG 0      /Origin of program is location 0
LDA A      /Load operand from location A
ADD B      /Add operand from location B
STA C      /Store sum in location C
HLT        /Halt computer
A,        DEC 83  /Decimal operand
B,        DEC -23 /Decimal operand
C,        DEC 0   /Sum stored in location C
END        /End of symbolic program
```

어셈블러

0010	0000	0000	0100
0001	0000	0000	0101
0011	0000	0000	0110
0111	0000	0000	0001
0000	0000	0101	0011
1111	1111	1110	1001
0000	0000	0000	0000

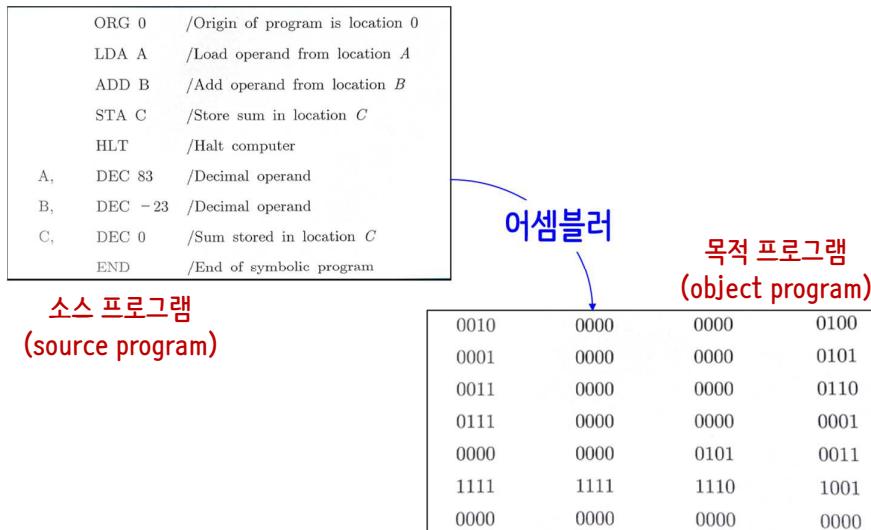


어셈블러, ASSEMBLER



어셈블러

- 어셈블리어 프로그램을 기계어로 번역하는 SW

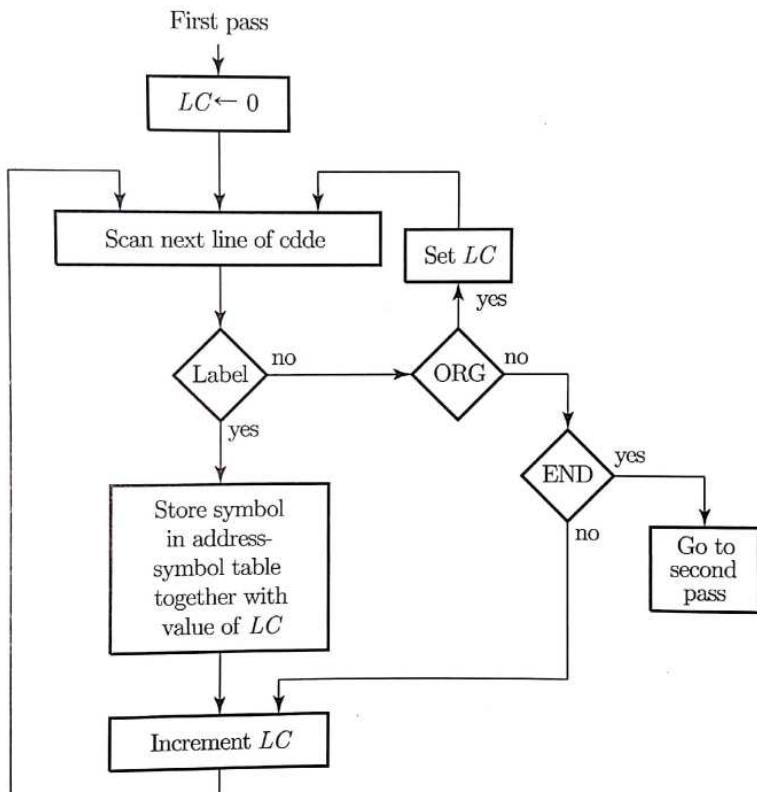


Two-Pass 어셈블러

- 소스 프로그램은 2번 스캔
- 1-Pass
 - 주소 기호(Label)과 그것의 주소를 결정
 - 심볼 테이블(symbol table, 주소-기호 테이블)을 작성
- 2-Pass
 - 소스 프로그램을 목적 프로그램으로 번역
 - 사용 테이블
 - 주소-기호 테이블
 - 명령어 테이블
 - 의사 명령어 테이블

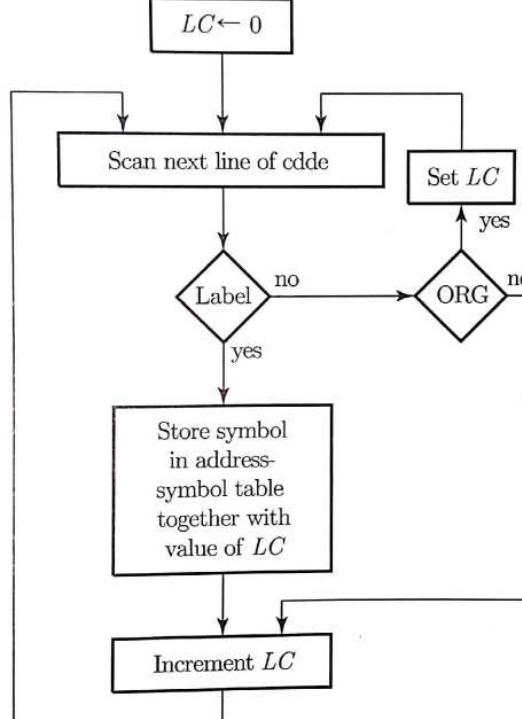


1-Pass 처리



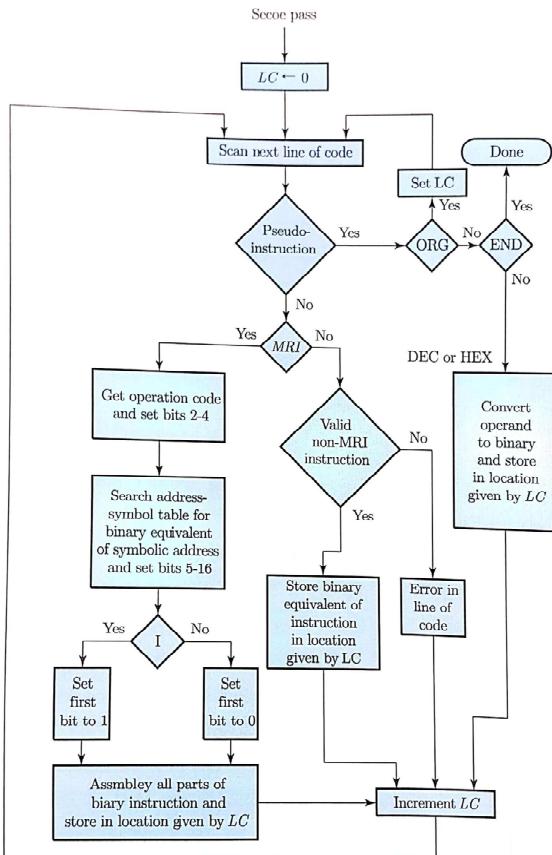
First pass

$LC \leftarrow 0$

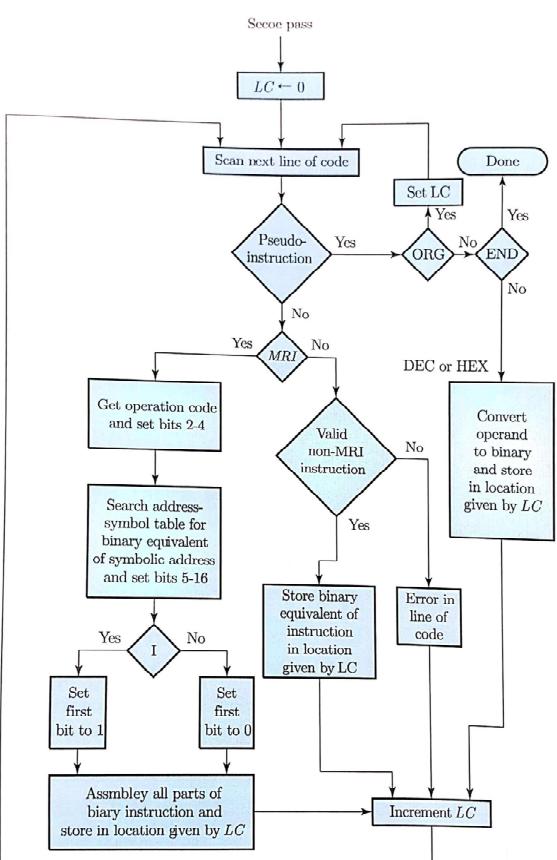


ORG 0	/Origin of program is location 0
LDA A	/Load operand from location A
ADD B	/Add operand from location B
STA C	/Store sum in location C
HLT	/Halt computer
A,	DEC 83 /Decimal operand
B,	DEC -23 /Decimal operand
C,	DEC 0 /Sum stored in location C
END	/End of symbolic program

2-Pass 처리



ORG 0	/Origin of program is location 0
LDA A	/Load operand from location A
ADD B	/Add operand from location B
STA C	/Store sum in location C
HLT	/Halt computer
A,	DEC 83 /Decimal operand
B,	DEC -23 /Decimal operand
C,	DEC 0 /Sum stored in location C
END	/End of symbolic program



서브루틴, SUBROUTINE



서브루틴, Subroutine

- 프로그램에서 여러 번 사용되는 공통된 명령어들
- 부프로그램 또는 함수
- 호출될 때 되돌아올 주소를 저장
 - 기본 컴퓨터 : **서브루틴 시작 위치**
 - 일반적으로는 스택(stack) 이용



서브루틴 호출의 예

Location		
100	ORG	100 /Main program
101	LDA	X /Load X
102	BSA	SH4 /Branch to subroutine
103	STA	X /Store shifted number
104	LDA	Y /Load Y
105	BSA	SH4 /Branch to subroutine again
106	STA	Y /Store shifted number
107		HLT
108		X, HEX 1234 Y, HEX 4321
109		SH4, HEX 0 /Subroutine to shift left 4 times
10A		CIL /Store return address here
10B		CIL /Circulate left once
10C		CIL
10D		CIL
10E		AND MSK /Circulate left fourth time
10F		BUN SH4 I /Set AC(13-16) to zero
110	MSK,	HEX FFF0 /Return to main program
		END /Mask operand

서브루틴 SH4 호출

서브루틴 SH4:
반환주소 저장

호출한 명령어의 다음
명령어로 분기;반환

서브루틴에 파라미터 전달

- 서브루틴 수행에 필요한 데이터를 전달
- 파라미터 전달 방법
 - 기본 컴퓨터 : **서브루틴 호출 명령어 다음 워드**
 - 일반적으로는 스택(stack) 이용

서브루틴에 파라미터 전달

Location	
200	ORG 200
201	LDA X
202	BSA OR
203	HEX 3AF6
204	STA Y
205	HLT
206	X, Y, OR,
207	HEX 7B95
208	HEX 0
209	HEX 0
20A	CMA
20B	STA TMP
20C	LDA OR I
20D	CMA
20E	AND TMP
20F	CMA
210	ISZ OR BUN OR I HEX 0 END



입출력 프로그램



입출력 방법

- INP 또는 OUT 명령어 이용하여 입, 출력
- 기본 컴퓨터의 FGI, FGO 값 검사가 요구됨
 - FGI(O) = 0 : 입(출)력 준비가 되지 않았음.
 - FGI(O) = 1 : 입(출)력 준비가 되어 정상 입(출)력 가능.

입출력 프로그램

(a) Input a character:	
CIF,	SKI
	BUN CIF
	INP
	OUT
	STA CHR
	HLT
CHR,	—
	CHR,
(b) Output one character:	
LDA CHR	
COF,	SKO
	BUN COF
	OUT
	HLT
CHR,	HEX 0057
	/Character is "W"

입출력이
준비될 때까지 계속
FGI 또는 FGO를 검사

→ 프로그램된 입출력(Programmed I/O)라고 함

인터럽트 입출력, Interrupt I/O

- 입출력 장치는 입출력이 완료되면 FGI 또는 FGO 값을 1로 설정.
- 기본 컴퓨터는 :
 - FGI 또는 FGO의 값이 1이고, IEN 값이 1일 때 인터럽트 처리
 - 다른 경우에는 입출력과 관련이 없는 작업 처리



인터럽트 처리

1. 레지스터들의 내용을 저장
2. FGI, FGO 값을 검사
3. FGI, FGO 값에 따라 입출력 처리
4. 저장해 둔 레지스터 값을 복구
5. 인터럽트 기능을 ON
6. 실행 중이던 프로그램으로 복귀



인터럽트 처리

	Location	인터럽트 처리 후 반환할 위치 저장
1. 레지스터 내용 저장	0 ZRO, 1 BUN SRV /Return address stored here	
2,3. 인터럽트 처리	100 CLA /Branch to service routine	
3. 인터럽트 처리 루틴으로 분기	101 ION /Portion of running program	
4. 레지스터 내용 복구	102 LDA X /Turn on interrupt facility	
5. 인터럽트 enable	103 ADD Y /Interrupt occurs here	
6. 인터럽트 처리 루틴에서 반환	104 STA Z /Program returns here after interrupt	
	105 . . .	
인터럽트 처리 루틴	200 SRV, STA SAC /Interrupt service routine	
	STA CIR /Store content of AC	
	STA SE /Move E into AC(1)	
	SKI /Store content of E	
	BUN NXT /Check input flag	
	INP /Flag is off, check next flag	
	OUT /Flag is on, input character	
	STA PT1 I /Print character	
	ISZ PT1 /Store it in input buffer	
	SKO /Increment input pointer	
	BUN EXT /Check output flag	
	LDA PT2 I /Flag is off, exit	
	OUT /Load character from output buffer	
	ISZ PT2 /Output character	
	LDA SE /Restore value of AC(1)	
	CIL /Shift it to E	
	LDA SAC /Restore content of AC	
	ION /Turn interrupt on	
	BUN ZRO I /Return to running program	
	SAC, — /AC is stored here	
	SE, — /E is stored here	
	PT1, — /Pointer of input buffer	
	PT2, — /Pointer of output buffer	

Q&A