Slide 0

6장. 기본 컴퓨터 프로그래밍

개요

- 컴퓨터 시스템의 구성
  - 하드웨어 : 컴퓨터를 구성하는 실제 장치들 (레지스터, CPU, RAM, ROM, ...)
  - 소프트웨어 : 컴퓨터를 동작 시키는 프로그램들 (OS, Compiler, user program, ...)
- 소프트웨어

Slide 1

  - 하드웨어에 종속적인 프로그램
    - 하드웨어에 따라 다르게 기술하여야 하는 프로그램 (low-level language)
    - 예) 기계어, 어셈블러
      기계어 (binary code 로 되어있음) →해당 컴퓨터로 실행가능
      어셈블리어 →기계어를 기호로 표현 →어셈블러 필요
  - 하드웨어에 비종속적인 프로그램
    - 하드웨어에 상관없이 기술할수 있는 프로그램 (high-level language)
    - 예) C, Fortran, Pascal, Basic, ...
      하드웨어에 종속적인 기계어로 바꾸는 컴파일러 필요

개요 (계속)

Slide 2

- 기본 컴퓨터의 25개의 명령어 (표 6-1)

표 6-1 컴퓨터 명령어

Symbol	Hexadecimal code	Description
AND	0 or 8	AND M to AC
ADD	1 or 9	Add M to AC, carry to E
LDA	2 or A	Load AC from M
STA	3 or B	Store AC in M
BUN	4 or C	Branch unconditionally to m
BSA	5 or D	Save return address in m and branch to $m + 1$
ISZ	6 or E	Increment M and skip if zero
CLA	7800	Clear AC
CLE	7400	Clear E
CMA	7200	Complement AC
CME	7100	Complement E
CIR	7080	Circulate right E and AC
CIL	7040	Circulate left E and AC
INC	7020	Increment AC,
SPA	7010	Skip if AC is positive
SNA	7008	Skip if AC is negative
SZA	7004	Skip if AC is zero
SZE	7002	Skip if E is zero
HLT	7001	Halt computer
INP	F800	Input information and clear flag
OUT	F400	Output information and clear flag
SKI	F200	Skip if input flag is on
SKO	F100	Skip if output flag is on
ION	F080	Turn interrupt on
IOF	F040	Turn interrupt off

기계어

Slide 3

- 프로그램
  - 컴퓨터로 하여금 특정 작업을 수행 시키기 위해 작성된 명령어 리스트
  - 종류
    1. 이진코드 : 실제로 실행될 형태의 프로그램 (기계어라고도 함)
    2. 8진수 또는 16진수 : 이진코드를 8진수나 16진수로 표현한것
    3. 기호코드 : 이진코드의 연산부분, 주소부분등을 기호로 표현 (어셈블리어라고도 함)
       * 어셈블러 : 기호 →이진코드
    4. 고급 프로그래밍 언어 : 하드웨어 종류에 상관없이 일을 중심으로 기술된 언어
       * 컴파일러 : 고급언어 →이진코드
  - 기본 컴퓨터에서의 언어
    * 두수를 가산하는 이진프로그램 (표 6-2)

기계어 (계속)

Slide 4

6.2 기계어 — 1

표 6-2 두 수를 가산하는 이진 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

→프로그래머가 이해하거나 프로그램하기 어려움

* 16진프로그램 으로 표현 (표 6-3)

기계어 (계속)

Slide 5

표 6-3 두 수를 가산하는 16진 프로그램

Location	Instruction
000	2004
001	1005
002	3006
003	7001
004	0053
005	FFE9
006	0000

→자릿수를 줄여서 나타냄

* 기호로 표현 (표 6-4)

기계어 (계속)

Slide 6

146 — 제6장 기본 컴퓨터 프로그래밍

표 6-4 기호 연산 코드를 가진 프로그램

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

대로 수행하지만 프로그램 작성자들은 이처럼 쓰여진 프로그램을 이해하기

* 어셈블리어로 표현 (표 6-5)

기계어 (계속)

Slide 7

6-5는 두 개의 숫자를 더하는 어셈블러 언어 프로그램이다. 기호 OR

표 6-5 두 수를 가산하는 어셈블리 언어 프로그램

```
      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
```

* 포트란으로 표현 (표 6-6)

기계어 (계속)

Slide 8

표 6-6 두 수를 가산하는 포트란 프로그램

```
INTEGER A, B, C
DATA A,83  B,-23
C = A + B
END
```

어셈블리 언어

Slide 9

- 프로그래밍 언어는 잘 정의된 일정한 규칙이 있어야함
- 기본컴퓨터의 어셈블리어 규칙
  - 구성
    1. 라벨 필드 : 기호 주소를 나타내거나 빈칸
    2. 명령어 필드 : 기계 명령어나 pseudo 명령어 기술
    3. 코멘트 필드 : 명령어에대한 주석이나 해설
  - 기호주소 : 3개이하의 영자숫자로 구성, 첫글자는 문자
  - 명령어
    1. 메모리 참조 명령어 (MRI)
    2. 레지스터 참조 또는 입출력 명령어 (non-MRI)
    3. pseudo 명령어
  - MRI
    * 표 6-1에 기술된 명령어
    * 기호주소
    * I : 있으면 간접주소 없으면 직접주소

어셈블리 언어 (계속)

* 예)

```
CLA non-MRI
ADD OPR 직접주소 MRI
ADD PTR I 간접주소 MRI
```

- Pseudo 명령어
  * 기계 명령어가 아니라 번역 (어셈블리어 →기계어) 에 필요한 명령어들 (표 6-7)

Slide 10

표 6-7 슈도 명령어의 정의

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

- comment : / 로 구분
- 예) 두수를 감산하는 어셈블리 언얼 프로그램 (표 6-8)

어셈블리 언어 (계속)

표 6-8 두 수를 감산하는 어셈블리 언어 프로그램

```
        ORG 100     /Origin of program is location 100
        LDA SUB     /Load subtrahend to AC
        CMA         /Complement AC
        INC         /Increment AC
        ADD MIN     /Add minuend to AC
        STA DIF     /Store difference
        HLT         /Halt computer
MIN,    DEC 83      /Minuend
SUB,    DEC -23     /Subtrahend
DIF,    HEX 0       /Difference stored here
        END         /End of symbolic program
```

Slide 11

- 이진수로의 변환
  * 어셈블러 : 어셈블리어를 기계어로 번역
  * 번역된 기계어 프로그램 (표 6-9)

어셈블리 언어 (계속)

Slide 12

표 6-9 표 6-8의 번역된 프로그램의 리스팅

Hexadecimal code			
Location	Content		Symbolic program
			ORG 100
100	2107		LDA SUB
101	7200		CMA
102	7020		INC
103	1106		ADD MIN
104	3108		STA DIF
105	7001		HLT
106	0053	MIN,	DEC 83
107	FFE9	SUB,	DEC −23
108	0000	DIF,	HEX 0
			END

* 기호프로그램의 두번 스캔 (scan)

1. 기계어 명령어와 피연산자의 메모리 영역만 지정 →주소-기호 테이블 생성

어셈블리 언어 (계속)

Slide 13

Address symbol	Hexadecimal address
MIN	106
SUB	107
DIF	108

2. 주소-기호 테이블을 이용 번역
3. 각각 first pass, second pass 라고함

어셈블러 (assembler)

Slide 14

- 어셈블리어 (source program) →기계어 (object program)
- 메모리내에서 기호 프로그램의 표현
  - 16진 문자 코드 (표 6-10)

어셈블러 (assembler) (계속)

Slide 15

표 6-10 16진 문자 코드

Character	Code	Character	Code	Character	Code	
A	41	Q	51	6	36	
B	42	R	52	7	37	
C	43	S	53	8	38	
D	44	T	54	9	39	
E	45	U	55	space	20	
F	46	V	56	(	28	
G	47	W	57	)	29	
H	48	X	58	*	2A	
I	49	Y	59	+	2B	
J	4A	Z	5A	,	2C	
K	4B	0	30	−	2D	
L	4C	1	31	.	2E	
M	4D	2	32	/	2F	
N	4E	3	33	=	3D	
O	4F	4	34	CR	0D	(carriage return)
P	50	5	35			

  - PD3, LDA SUB I 란 명령어의 컴퓨터 표현 (표 6-11)

어셈블러 (assembler) (계속)

Slide 16

152 제6장 기본 컴퓨터 프로그래밍

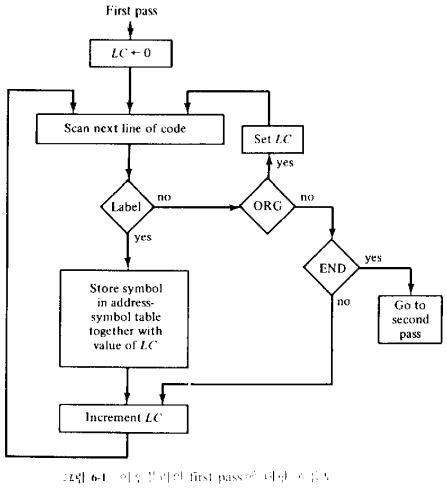
표 6-11 PL3, LDA SUB I이란 코드의 컴퓨터 표현

Memory word	Symbol	Hexadecimal code	Binary representation
1	P L	50 4C	0101 0000 0100 1100
2	3 ,	33 2C	0011 0011 0010 1100
3	L D	4C 44	0100 1100 0100 0100
4	A	41 20	0100 0001 0010 0000
5	S U	53 55	0101 0011 0101 0101
6	B	42 20	0100 0010 0010 0000
7	I CR	49 0D	0100 1001 0000 1101

- First pass
  - 사용자가 정의한 주소기호와 저장위치간의 관계를 표로 작성
  - LC (location counter) 사용
  - first pass 의 흐름도 (그림 6-1)

어셈블러 (assembler) (계속)

Slide 17



- 표 6-8의 프로그램에 대한 주소-기호 테이블 (표 6-12)

어셈블러 (assembler) (계속)

Slide 18

표 6-12 표 6-8의 프로그램에 대한 주소-기호 테이블

Memory word	Symbol or (LC)*	Hexadecimal code	Binary representation
1	M I	4D 49	0100 1101 0100 1001
2	N ,	4E 2C	0100 1110 0010 1100
3	(LC)	01 06	0000 0001 0000 0110
4	S U	53 55	0101 0011 0101 0101
5	B ,	42 2C	0100 0010 0010 1100
6	(LC)	01 07	0000 0001 0000 0111
7	D I	44 49	0100 0100 0100 1001
8	F ,	46 2C	0100 0110 0010 1100
9	(LC)	01 08	0000 0001 0000 1000

* (LC) designates content of location counter.

- Second pass
  - 4개의 테이블을 이용 번역
    1. pseudo 명령어 테이블
    2. MRI 테이블
    3. Non-MRI 테이블
    4. 주소-기호 테이블

어셈블러 (assembler) (계속)

Slide 19

- second pass 의 흐름도 (그림 6-2)

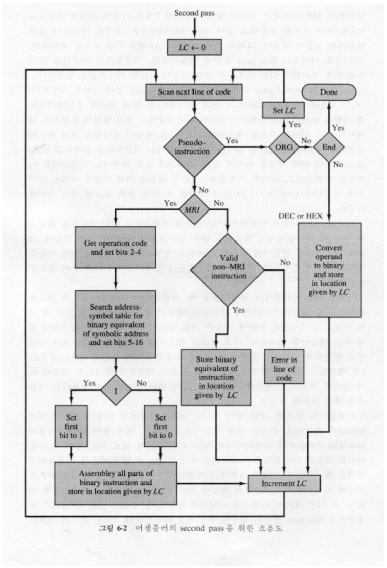


그림 6-2 어셈블러의 second pass를 위한 흐름도

프로그램 루프

- 컴파일러는 중간단계로 어셈블리어로 변환하기도 하고 직접 기계어로 번역하기도 함

Slide 20

- 100개의 수를 더하는 포트란 프로그램

```
   DIMMENSION A(100)
   INTEGER SUM,A
   SUM = 0
   DO 3 J=1,100
3  SUM = SUM + A(j)
```

- 포트란 프로그램을 컴파일한 결과로 출력되는 어셈블리어 프로그램 (표 6-13)

프로그램 루프 (계속)

Slide 21

표 6-13 100개의 수를 가산하는 기호 프로그램

Line			
1		ORG 100	/Origin of program is HEX 100
2		LDA ADS	/Load first address of operands
3		STA PTR	/Store in pointer
4		LDA NBR	/Load minus 100
5		STA CTR	/Store in counter
6		CLA	/Clear accumulator
7	LOP,	ADD PTR I	/Add an operand to AC
8		ISZ PTR	/Increment pointer
9		ISZ CTR	/Increment counter
10		BUN LOP	/Repeat loop again
11		STA SUM	/Store sum
12		HLT	/Halt
13	ADS,	HEX 150	/First address of operands
14	PTR,	HEX 0	/This location reserved for a pointer
15	NBR,	DEC −100	/Constant to initialized counter
16	CTR,	HEX 0	/This location reserved for a counter
17	SUM,	HEX 0	/Sum is stored here
18		ORG 150	/Origin of operands is HEX 150
19		DEC 75	/First operand
•			
•			
•			
118		DEC 23	/Last operand
119		END	/End of symbolic program

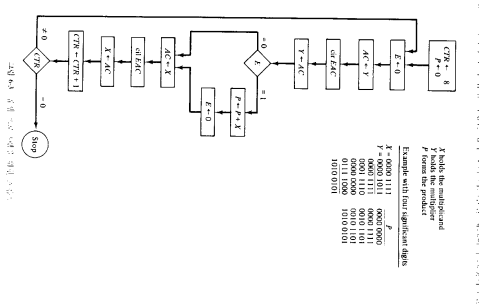
산술 및 논리 연산의 프로그래밍

Slide 22

- 명령어 세트에 주어지지 않은 명령의 실행을 위해서는 프로그램으로 해결
- 곱셈 프로그램
  - 가정
    1. 부호비트 무시 양수만 가정
    2. 두수는 8비트로서 곱은 16비트 초과하지 않음
  - 곱셈 프로그램을 위한 흐름도 (그림 6-3)

산술 및 논리 연산의 프로그래밍 (계속)

Slide 23



  - 곱셈 프로그램 (표 6-14)

산술 및 논리 연산의 프로그래밍 (계속)

Slide 24

표 6-14 두 개의 양수를 곱하는 프로그램

```
        ORG 100
LOP,    CLE         /Clear E
        LDA Y       /Load multiplier
        CIR         /Transfer multiplier bit to E
        STA Y       /Store shifted multiplier
        SZE         /Check if bit is zero
        BUN ONE     /Bit is one; go to ONE
        BUN ZRO     /Bit is zero; go to ZRO
ONE,    LDA X       /Load multiplicand
        ADD P       /Add to partial product
        STA P       /Store partial product
        CLE         /Clear E
ZRO,    LDA X       /Load multiplicand
        CIL         /Shift left
        STA X       /Store shifted multiplicand
        ISZ CTR     /Increment counter
        BUN LOP     /Counter not zero; repeat loop
        HLT         /Counter is zero; halt
CTR,    DEC -8      /This location serves as a counter
X,      HEX 000F    /Multiplicand stored here
Y,      HEX 000B    /Multiplier stored here
P,      HEX 0       /Product formed here
        END
```

- 배 정밀도 가산
  - 두개의 16비트를 곱할면 결과는 32 비트 →두개의 메모리 워드에 저장

산술 및 논리 연산의 프로그래밍 (계속)

Slide 25

→배정밀도

- 배정밀도 가산 필요 →프로그램 (표 6-15)

표 6-15 두 개의 배정밀도를 가산하는 프로그램

```
        LDA AL      /Load A low
        ADD BL      /Add B low, carry in E
        STA CL      /Store in C low
        CLA         /Clear AC
        CIL         /Circulate to bring carry into AC(16)
        ADD AH      /Add A high and carry
        ADD BH      /Add B high
        STA CH      /Store in C high
        HLT
AL,     —           /Location of operands
AH,     —
BL,     —
BH,     —
CL,     —
CH,     —
```

- 논리연산
  - 기본컴퓨터에서 가능한 연산 →AND, CMA, CLA

산술 및 논리 연산의 프로그래밍 (계속)

Slide 26

- 다른 연산은 프로그램으로 가능
- 예) 논리 OR
  - DeMorgan 정리에의해 계산 →A + B = (A'B')'

```
LDA A /Load first operand A
CMA /Complement to get A'
STA TMP /Store in a temporary location
LDA B /Load second operand B
CMA /Complement to get B'
AND TMP /AND with A' to get A' AND B'
CMA /Complement again to get A OR B
```

- 시프트 연산
  - 순환시프트외에 논리 시프트와 산술 시프트가 있음
  - 기본컴퓨터에서는 순환시프트만 있음

    →논리 및 산술 시프트는 순환시프트로 프로그램

산술 및 논리 연산의 프로그래밍 (계속)

Slide 27

- 논리 오른쪽 시프트

```
CLE
CIR
```

- 논리 왼쪽 시프트

```
CLE
CIL
```

- 산술 오른쪽 시프트
  - 부호비트는 고정되어야함

```
CLE /Clear E to 0
SPA /Skip if AC is positive; E remains 0
CME /AC is negative, set E to 1
CIR /Circulate E and AC
```

- 산술 왼쪽 시프트
  - 부호비트는 고정되어야함
  - 논리 왼쪽 시프트와 등일하나 시프트후 E 와 msb 비교

    동일하면 정상적동작, 다르면 오우버플로우

서브루틴

- 프로그램에서 여러번 사용되는 공통의 명령어들을 서브루틴이라함
- 주프로그램의 어느곳에서도 서브루틴으로 분기 가능
- 서브루틴의 사용예 (표 6-16)

Slide 28

6.7 서브루틴 — 165

표 6-16 서브루틴의 사용을 보이는 프로그램

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

서브루틴 (계속)

- 서브루틴 파라메터와 데이타 링키지

Slide 29

  - 표 6-16 에서 파라메터는 AC 를 통하여 전달됨
  - 다수의 파라메터는 레지스터나 메모리를 통하여 전달
  - 예) 표 6-17
  - 100개의 데이타를 100번지에서 200번지로 이동시키는 서브루틴 (표 6-18)

서브루틴 (계속)

Slide 30

표 6-18 데이타의 블럭을 이동시키는 서브루틴

```
                    /Main program
        BSA MVE     /Branch to subroutine
        HEX 100     /First address of source data
        HEX 200     /First address of destination data
        DEC -16     /Number of items to move
        HLT
MVE,    HEX 0       /Subroutine MVE
        LDA MVE I   /Bring address of source
        STA PT1     /Store in first pointer
        ISZ MVE     /Increment return address
        LDA MVE I   /Bring address of destination
        STA PT2     /Store in second pointer
        ISZ MVE     /Increment return address
        LDA MVE I   /Bring number of items
        STA CTR     /Store in counter
        ISZ MVE     /Increment return address
LOP,    LDA PT1 I   /Load source item
        STA PT2 I   /Store in destination
        ISZ PT1     /Increment source pointer
        ISZ PT2     /Increment destination pointer
        ISZ CTR     /Increment counter
        BUN LOP     /Repeat 16 times
        BUN MVE I   /Return to main program
PT1,    —
PT2,    —
CTR,    —
```

입출력 프로그래밍

- 한 문자를 입출력 시키는 프로그램 (표 6-19)

Slide 31

표 6-19 한 문자를 입출력시키는 프로그램

```
(a) Input a character:
CIF,    SKI         /Check input flag
        BUN CIF     /Flag=0, branch to check again
        INP         /Flag=1, input character
        OUT         /Print character
        STA CHR     /Store character
        HLT
CHR,    —           /Store character here
(b) Output one character:
        LDA CHR     /Load character into AC
COF,    SKO         /Check output flag
        BUN COF     /Flag=0, branch to check again
        OUT         /Flag=1, output character
        HLT
CHR,    HEX 0057    /Character is "W"
```

- 문자 조작

입출력 프로그래밍 (계속)

Slide 32

- 두개의 문자 (8비트 + 8비트) 를 묶는 프로그램
- 두문자를 입력하여 묶는 서브루틴 (표 6-20)

표 6-20 두 문자를 입력하여 패킹시키는 서브루틴

```
IN2,    —          /Subroutine entry
FST,    SKI
        BUN FST
        INP        /Input first character
        OUT
        BSA SH4    /Shift left four times
        BSA SH4    /Shift left four more times
SCD,    SKI
        BUN SCD
        INP        /Input second character
        OUT
        BUN IN2 I  /Return
```

- 버퍼에 입력 문자를 저장시키는 프로그램 (표 6-21)

입출력 프로그래밍 (계속)

Slide 33

표 6-21 버퍼에 입력 문자를 저장시키는 프로그램

```
        LDA ADS    /Load first address of buffer
        STA PTR    /Initialize pointer
LOP,    BSA IN2    /Go to subroutine IN2 (Table 6-20)
        STA PTR I  /Store double character word in buffer
        ISZ PTR    /Increment pointer
        BUN LOP    /Branch to input more characters
        HLT
ADS,    HEX 500    /First address of buffer
PTR,    HEX 0      /Location for pointer
```

- 두 워드를 비교하는 프로그램 (표 6-22)

입출력 프로그래밍 (계속)

Slide 34

표 6-22 두 워드를 비교하는 프로그램

```
        LDA WD1     /Load first word
        CMA
        INC         /Form 2's complement
        ADD WD2     /Add second word
        SZA         /Skip if AC is zero
        BUN UEQ     /Branch to "unequal" routine
        BUN EQL     /Branch to "equal" routine
WD1,    —
WD2,    —
```

- 프로그램 인터럽트
  - 프로그램 실행의 효율성을 증대시킴
  - 인터럽트를 서비스하는 프로그램 예 (표 6-23)

Homework

Slide 35

- 4장 연습문제
  - 1, 3, 7, 11, 19
- 5장 연습문제
  - 2, 4, 9, 20
- 6장 연습문제
  - 2, 3, 14