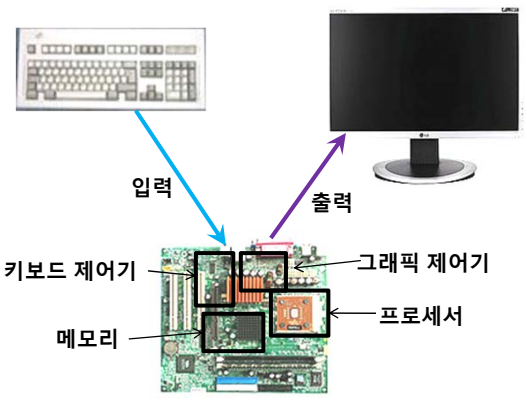
© Yong-Seok Kim, Kangwon National University, 2012~9

제7장
입출력 장치 제어와 운영 체제 기능 호출

Youtube 주소
[7-1] http://http://youtu.be/NnGoxruAcI8
[7-2] http://youtu.be/apaoijBZb1s
[7-2(추가)] http://youtu.be/GXXG5TdUpDY

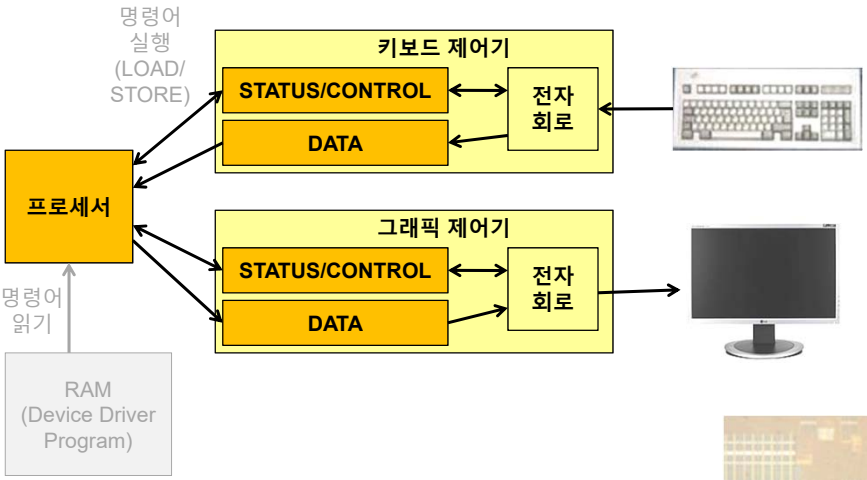
© Yong-Seok Kim, Kangwon National University. 2012~9

프로세서와 입출력 장치



© Yong-Seok Kim, Kangwon National University. 2012~9

입출력 장치 제어기



© Yong-Seok Kim, Kangwon National University. 2012~9

입출력장치 레지스터의 주소

레지스터 이름	레지스터 주소	용도
KBDATA	0xFFFFFF00	키보드의 눌려진 문자 보관
KBSC	0xFFFFFF01	키보드 상태/제어
DPDATA	0xFFFFFF02	화면에 출력할 문자 기록
DPSC	0xFFFFFF03	화면 상태/제어

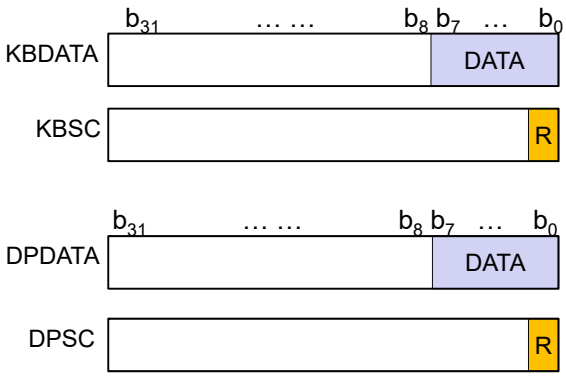
© Yong-Seok Kim, Kangwon National University. 2012~9

메모리 맵 예

이름	주소	용도
RAM	0x00000000 ~ 0x7FFFFFFF	RAM 영역 (2G)
ROM	0xE0000000 ~ 0xEFFFFFFF	ROM 영역 (256M)
KBDATA	0xFFFFFF00	키보드의 눌려진 문자 보관
KBSC	0xFFFFFF01	키보드 상태/제어
DPDATA	0xFFFFFF02	화면에 출력할 문자 기록
DPSC	0xFFFFFF03	화면 상태/제어

© Yong-Seok Kim, Kangwon National University. 2012~9

입출력 장치 레지스터



© Yong-Seok Kim, Kangwon National University. 2012~9

키보드 장치 구동

- (가정) 키보드 제어기
  - ✓키가 눌려지면 KBDATA 레지스터에 문자코드를 기록하고 KBSC 레지스터의 R 비트를 1로 설정
  - ✓R 비트가 1인 상태에서는 키를 눌러도 무시
- 키보드 문자 읽기 (디바이스 드라이버 프로그램)
  - ✓KBSC 를 읽어서 R 비트 검사
  - ✓R 비트가 0이면 KBSC를 읽고 검사하기를 반복
  - ✓R 비트가 1이면 KBDATA를 읽어서 R0에 저장하고 KBSC의 R 비트는 0으로 재설정

© Yong-Seok Kim, Kangwon National University. 2012~9

키보드 문자 입력 함수

```
GetChar:
    LOAD R1, KBSC       ; 키보드 제어 레지스터 주소
gcL0: LDR R0, R1, 0     ; 키보드 상태 읽기
    AND R0, R0, 1       ; 최하위 비트 검사를 위해
    BR z, gcL0          ; 최하위 비트가 0이면 다시 gcL0로 이동
    LOAD R2, KBDATA     ; 키보드 데이터 레지스터 주소
    LDR R0, R2, 0       ; 키보드 문자 읽기
    COPY R2, 0          ; R2의 최하위 비트를 0으로 하여
    STR R2, R1, 0       ; 키보드 제어 레지스터에 기록
    RTI
KBDATA: .FILL 0xFFFFFF00
KBSC:   .FILL 0xFFFFFF01
```

© Yong-Seok Kim, Kangwon National University. 2012~9

디스플레이 장치 구동

- (가정) 그래픽 제어기
  - ✓DPSC 레지스터을 읽어서 R비트 가 1이면 DPDATA 레지스터의 문자를 화면의 현재 커서위치에 추가하고 R 비트를 0으로 재설정
  - ✓R 비트가 0이면 현재의 화면표시 유지
- 화면에 문자 쓰기(디바이스 드라이버 프로그램)
  - ✓DPSC 를 읽어서 R 비트 검사
  - ✓R 비트가 1이면 DPSC를 읽고 검사하기를 반복
  - ✓R 비트가 0이면 DPDATA에 R0의 값을 저장하고, DPSC의 R 비트는 1로 설정

© Yong-Seok Kim, Kangwon National University. 2012~9

화면에 문자 출력 함수

```
PutChar:
    LOAD R1, DPSC        ; 화면 제어 레지스터 주소
pcL0: LDR R2, R1, 0      ; 화면 상태 읽기
    AND R2, R2, 1        ; 최하위 비트 검사를 위해
    BR p, pcL0           ; 최하위 비트가 1이면 다시 pcL0로 이동
    LOAD R2, DPDATA      ; 화면 데이터 레지스터 주소
    STR R0, R2, 0        ; 화면에 문자 쓰기
    COPY R2, 1           ; R2의 최하위 비트를 1로 하여
    STR R2, R1, 0        ; 화면 제어 레지스터에 기록
    RTI
DPDATA: .FILL 0xFFFFFF02
DPSC:   .FILL 0xFFFFFF03
```

© Yong-Seok Kim, Kangwon National University. 2012~9

운영체제 기능 호출을 위한 명령어

- 시스템 호출 (system call)
  - ✓운영체제에서 제공하는 특정 기능의 실행을 요청
  - ✓운영체제마다 고유의 시스템 호출 기능 집합 제공
- 시스템 호출 예
  - ✓키보드 문자 읽기, 화면에 문자 출력, 프로그램 종료
  - ✓파일 열기, 파일 읽기, 파일 쓰기, 파일 닫기, …
- 시스템 호출을 위한 명령어

이름	어셈블리언어 표현	의미
Software Interrupt	`SWI n`	운영체제의 n 번째 기능 호출
Return from Interrupt	`RTI`	운영체제에서 응용 프로그램으로 복귀

© Yong-Seok Kim, Kangwon National University. 2012~9

운영체제의 기능 호출

이름	어셈블리언어 표현	의미
문자 읽기	**`SWI 0`**	키보드에서 한 문자를 읽어서 R0에 기록
문자 쓰기	**`SWI 1`**	R0에 저장된 문자를 화면에 출력
문자열 읽기	**`SWI 2`**	키보드에서 문자열을 읽어서 R0가 가리키는 메모리 주소부터 차례대로 저장, 문자열의 끝은 0(null) 로 처리
문자열 쓰기	**`SWI 3`**	R0가 가리키는 메모리 주소로부터 시작되는 문자열을 화면에 출력, 문자열의 끝은 0(null) 문자로 가정
에코모드 설정	**`SWI 4`**	키보드 입력시의 에코모드 설정: R0의 값이 1이면 에코 실행, R0의 값이 0이면 에코 없음 (기본 모드는 에코 있음)
프로그램 종료	**`SWI 255`**	실행중인 프로그램의 종료

© Yong-Seok Kim, Kangwon National University. 2012~9

에코 처리가 있는 키보드 문자 입력 함수

```
GetChar:
    LOAD R1, KBSC
gcL0: LDR R0, R1, 0
    AND R0, R0, 1
    BR z, gcL0
    LOAD R2, KBDATA
    LDR R0, R2, 0
    COPY R2, 0
    STR R2, R1, 0
    LOAD R1, echoMode
    CMP R1, 0
    BR np, PutChar
    RTI
KBDATA: .FILL 0xFFFFFF00
KBSC:   .FILL 0xFFFFFF01
SetEcho:
    STORE R0, echoMode
    RTI
echoMode: .FILL 1
```

```
PutChar:
    LOAD R1, DPSC
pcL0: LDR R2, R1, 0
    AND R2, R2, 1
    BR p, pcL0
    LOAD R2, DPDATA
    STR R0, R2, 0
    COPY R2, 1
    STR R2, R1, 0
    RTI
DPDATA: .FILL 0xFFFFFF02
DPSC:   .FILL 0xFFFFFF03
```


© Yong-Seok Kim, Kangwon National University. 2012~9

main 함수의 시작과 종료의 처리

```
    LEA R5, _END          ; R5에 스택 영역 상단 주소 기록
    ADD R5, R5, 1024      ; R5는 스택 영역 바닥 주소로 변경
    LINK
    BR nzp, main          ; main 함수로 이동
    SWI 255               ; main에서 복귀하면 프로그램 종료
main:       ; main 함수
    ...
    RET

    ...

_END:                     ; 스택 메모리 영역 상단 주소
    .BLOCK 1024           ; 영역 크기 1024
```

© Yong-Seok Kim, Kangwon National University. 2012~9

TOY C 컴파일러의 라이브러리 함수

C 언어 표현	사용 예	의미
getchar(void)	char ch; ch = getchar();	키보드에서 한 문자를 읽어서 변수 ch에 기록
putchar(char)	putchar(ch); putchar('A');	인수에 지정한 문자를 화면에 출력
gets(char[])	char a[20]; gets(a);	키보드에서 문자열을 읽어서 인수에 지정한 배열의 시작부터 차례대로 저장, 문자열의 끝은 null (ASCII 0) 문자로 처리
puts(char[])	puts(a); puts("Yes");	인수에 지정한 배열 (또는 문자열)의 시작부터 null (ASCII 0) 문자가 나올 때까지 차례대로 화면에 출력
setecho(int)	setecho(0); setecho(1);	키보드 입력시 에코 없이(0), 또는 에코가 출력되게(1) 설정
exit(void)	exit();	호출한 프로그램을 종료

© Yong-Seok Kim, Kangwon National University. 2012~9

TOY C 컴파일러의 라이브러리 함수 컴파일

C 언어 표현	어셈블리 언어 출력
getchar(void)	getchar: SWI 0 RET
putchar(char)	putchar: LDR R0, R5, 0 SWI 1 RET
gets(char[])	gets: LDR R0, R5, 0 SWI 2 RET
puts(char[])	Puts: LDR R0, R5, 0 SWI 3 RET
setecho(int)	setecho: LDR R0, R5, 0 SWI 4 RET
exit(void)	exit: SWI 255 RET

© Yong-Seok Kim, Kangwon National University. 2012~9

(추가) 문자열 입출력 활용 예

```
// print in reverse order
char word[100];
char word2[100];
int main()
{
    while (1) {
        puts("input: ");
        gets(word);
        reverse();
        puts(word2);
        putchar('\n');
    }
}
```

```
int reverse()
{   int i, i2;

    i = 0;
    while (word[i] != 0)
        i = i + 1;

    i2 = 0;
    while (i > 0) {
        i = i - 1;
        word2[i2] = word[i];
        i2 = i2 + 1;
    }
    word2[i2] = 0;
}
```


© Yong-Seok Kim, Kangwon National University. 2012~9

정수 입력/출력 함수

```
int getint()
{
   int num;  char ch;

   num = 0;
   while (1) {
     ch = getchar();
     if (ch < '0')
       return num;
     if (ch > '9')
       return num;
    num = num * 10 + ch - '0';
   }
}
```

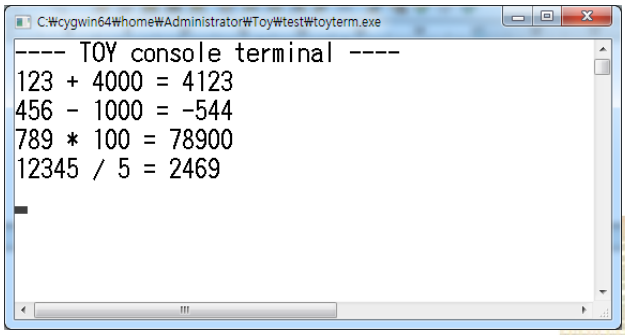
```
putint(int num)
{
  int quot;

   if (num < 0) {
       putchar('-');
       num = -num;
   }
   quot = num / 10;
   if (quot != 0)
       putint(quot);
   putchar(num % 10 + '0');
}
```

© Yong-Seok Kim, Kangwon National University. 2012~9

간단한 계산기 (calc.c)

- C 프로그램 : calc.c (프로그램 7.6)
- TOY C 컴파일러 출력: calc.s
- Toyas 출력: calc.o
- Toysim으로 실행 (toyterm을 통해서 입출력)
- Toysim + toyterm 대신에 toyvm 만 실행해도 됨



© Yong-Seok Kim, Kangwon National University. 2012~9

운영체제의 시스템 호출 처리

```
  .ORIGIN 0       ; 시작 주소를 0번지로
  .FILL GetChar   ; SWI 0 (키보드에서 1문자 읽기)
  .FILL PutChar   ; SWI 1 (화면에 1문자 출력)
  .FILL GetS      ; SWI 2 (키보드에서 문자열 읽기)
  .FILL PutS      ; SWI 3 (화면에 문자열 출력)
  .FILL SetEcho   ; SWI 4 (에코모드 설정)
  .BLOCK 250      ; 나머지 250개를 위한 영역
  .FILL Exit      ; SWI 255 (프로그램 종료)

GetChar: ; 키보드에서 1문자 읽기
    ...  ; (프로그램 7.3 참조)
    RTI  ; SWI 를 실행한 위치로 복귀
KBDATA: .FILL 0xFFFFFF00
KBSC:   .FILL 0xFFFFFF01
PutChar: ; 화면에 1문자 출력
    ...  ; (프로그램 7.2 참조)
    RTI
DPDATA: .FILL 0xFFFFFF02
DPSC:   .FILL 0xFFFFFF03
```

```
GetS:    ; 키보드에서 문자열 읽기
    ...
    RTI
PutS:    ; 화면에 문자열 출력
    ...
    RTI
SetEcho: ; 에코모드 설정
    ...
    RTI
echoMode: .FILL 1   ; 에코적용
Exit:    ; 호출한 프로그램의 종료
    ...
    RTI
```