# TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-04-19 (40 회차)

강사: Innova Lee(이상훈)
gcccompil3r@gmail.com
학생: 정유경
ucong@naver.com

1. screen 설치
ctrl ac : 가상터미널 생성
ctrl aa : 터미널 변경

2. 카페 → 인텔 시스템 프로그래밍 문서
https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-system-programming-manual-325384.pdf

**CR0** (제어 레지스터)

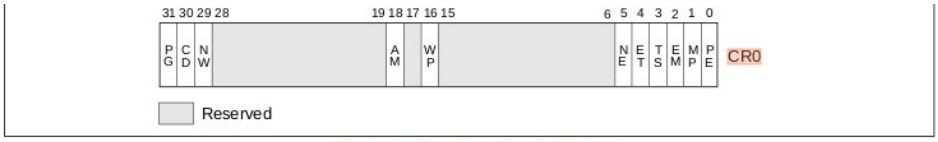

Figure 2-7. Control Registers

→ 2.1.6 System Registers
To assist in initializing the processor and controlling system operations, the system architecture provides system flags in the EFLAGS register and several system registers:
The **control registers (CR0**, CR2, CR3, and CR4) contain a variety of flags and data fields for controlling system-level operations.

→ 2.5 CONTROL REGISTERS
Control registers (CR0, CR1, CR2, CR3, and CR4; see Figure 2-7) determine operating mode of the processor and the characteristics of the currently executing task.
CR0 — Contains system control flags that control **operating mode and states of the processor**.
(즉, 프로세서의 상태와 시스템 동작모드 설정)

PG Paging (bit 31 of CR0) — Enables paging when set;
(1: 가상메모리를 물리메로리로 바꾼다)
처음부터활성화 시키지 않는이유 : 처음부팅할때 어셈블리어가 동작한다
이때 가상메모리 설정이 되어 있지 않다. 꺼놓고 있다가, 메모리 레이아웃 잡고 나서 켠다.

CD Cache Disable (bit 30 of CR0) — When the CD flag is set, caching is restricted

**WP** Write Protect (bit 16 of CR0) — **When set, inhibits** supervisor-level procedures from **writing** into read-only pages; **when clear, allows** supervisor-level procedures to write into read-only pages (regardless of the U/S bit setting; see Section 4.1.3 and Section 4.6). This flag facilitates implementation of the **copy-on-write(구현)** method of creating a new process (forking) used by operating systems such as UNIX.

3. monitor_hack.c 코드분석

```
 int syscall_hooking_init(void)
{
	unsigned long cr0;

	if((sys_call_table = locate_sys_call_table()) == NULL) // 시스템 콜 테이블을 찾는다
	{
		printk("<0>Can't find sys_call_table\n");
		return -1;
	}

	printk("<0>sys_call_table is at[%p]\n", sys_call_table);

	// CR0 레지스터를 읽어옴
	cr0 = read_cr0(); // read_cr0() → 3 → native_read_cr0(void)
```

```
static inline unsigned long native_read_cr0(void)
{
	unsigned long val;
//  asm volatile 구동시킬 명령어 : 출력 : 입력 : 어셈블러 지시어
	asm volatile("mov %%cr0,%0\n\t" : "=r" (val), "=m" (__force_order)); // r: 레지스터 m: 메모리
// mov cr0 값을 val 에 넣는다  // %0(맨 처음 나타난 변수, val)
// 함수에 입력이 없으므로 그냥, cr0 레지스터를 통해 val 를 출력
	return val; // cr0 의 값이 val 로 전달
}
```

인텔 문서에서 cr0 찾아보기

```
	// Page 쓰기를 허용함
	write_cr0(cr0 & ~0x00010000);
```

```
 & ~0x00010000 : 0~15 번 비트는 0, 16 번째 비트는 1 인데 ~
→ WP(16 번 bit) 를 0 으로 만든다 → allows write into read-only pages
물리메모리에 접근해서 내용을 바꿀 수 있다
```

```
 write_cr0 → 3 →  native_write_cr0(x)
static inline void native_write_cr0(unsigned long val)
{
	asm volatile("mov %0,%%cr0": : "r" (val), "m" (__force_order));
// %0 는 val, val 로 넘어온 값(cr0 & ~0x00010000)을 cr0 에 넣는다 → 페이지 쓰기를 허용한다
}
```

```
 /* set_memory_rw 라는 심볼을 찾아와서 fixed_set_memory_rw 에 >설정함 */
	fixed_set_memory_rw = (void *)kallsyms_lookup_name("set_memory_rw");
// set_memory_rw 를 하기위해서 심볼을 찾는다, 메모리를 읽고 쓰기위해서, 즉 찾지못하면 심볼을 쓸 수 없다
즉, 메모리를 읽고 쓸수 없다
// 결과적으로  fixed_set_memory_rw 에 set_memory_rw 가 셋팅된다
```

```
1 번은 return 0 의미없다
2 번을 본다
/* Lookup the address for this symbol. Returns 0 if not found. */
```

```
// 심볼에 대해 주소를 검색한다. 찾지 못하면 0 을 리턴한다
unsigned long kallsyms_lookup_name(const char *name)
{
	char namebuf[KSYM_NAME_LEN];
	unsigned long i;
	unsigned int off;

	for (i = 0, off = 0; i < kallsyms_num_syms; i++) {
		off = kallsyms_expand_symbol(off, namebuf, ARRAY_SIZE(namebuf));
//  kallsyms_expand_symbol 커널내 전체 심볼을 검색한다
// off: 오프셋, namebuf : 배열이름, ARRAY_SIZE(namebuf): 배열의 크기

		if (strcmp(namebuf, name) == 0)
			return kallsyms_addresses[i];
	}
	return module_kallsyms_lookup_name(name);
}
```

```
kallsyms_lookup_name → ,css 1

extern const unsigned long kallsyms_num_syms
//extern : 외부에서 참조한다. 즉 어딘가에 정의되어 있다→ E257: cstag: tag not found, #define 해서 매크로 써서 이상하게 되어있다, 알수있는 방법이 없다!
__attribute__((weak, section(".rodata")));
```

```
kallsyms_expand_symbol

static unsigned int kallsyms_expand_symbol(unsigned int off,
                                           char *result, size_t maxlen)
{
	int len, skipped_first = 0;
	const u8 *tptr, *data;

	/* Get the compressed symbol length from the first symbol byte. */
	data = &kallsyms_names[off]; // 전달인자 offset 은 현재 0,첫번째 배열의 인덱스가 data 로 들어간다
	len = *data; // 그걸 포인터로 받아주면 길이가 나온다
	data++;

	/*
	 * Update the offset to return the offset for the next symbol on
	 * the compressed stream.
	 */
	off += len + 1; // 길이값 만큼 offset 을 늘려준다
	/*
	 * For every byte on the compressed symbol data, copy the table
	 * entry for that byte.
	 */
	while (len) { //루프돌면서 심볼값이 있는지 찾는다, 있으면 해당 오프셋 위치를 리턴
/*만일 namebuf 와 name 이 일치하면 0 이다? 즉, 찾은 심볼을 리턴한다, index 만큼 루프를 돈다, 찾지 못하면 다시 루프를 돌면서 index 를 하나씩 증가시킨다*/
		tptr = &kallsyms_token_table[kallsyms_token_index[*data]];
```

```
            data++;
            len--;
// 그래도 없으면 만든 모듈, 즉 드라이버? 중에 심볼이 있는지 찾는다.
            while (*tptr) {
                  if (skipped_first) {
                        if (maxlen <= 1)
                              goto tail;
                        *result = *tptr;
                        result++;
                        maxlen--;
                  } else
                        skipped_first = 1;
                  tptr++;
            }
      }

tail:
 if (maxlen)
            *result = '\0';

      /* Return to offset to the next symbol. ??*/
      return off;
}
```

```
5
int set_memory_rw(unsigned long addr, int numpages) // 주소와 페이지 개수
{
      return change_page_attr_set(&addr, numpages, __pgprot(_PAGE_RW), 0);
}
```

```
      if(!fixed_set_memory_rw)
      {
            printk("<0>Unable to find set_memory_rw symbol\n");
            return 0;
      }

      /* 시스템 콜 테이블이 위치한 물리 메모리에 읽고 쓰기 권한 주기
 */
      fixed_set_memory_rw(PAGE_ALIGN((unsigned long)sys_call_table) - PAGE_SIZE, 3);
// PAGE_ALIGN 해서 syscalltable 을 페이지 단위로 정렬시키고, 페이지 사이즈를 빼주고 전달인자 3 을 넘긴다
```

```
5
int set_memory_rw(unsigned long addr, int numpages) // 주소와 페이지 개수
{
      return change_page_attr_set(&addr, numpages, __pgprot(_PAGE_RW), 0);
//change_page_attr_set : 페이지 속성을 설정한다.
}
```

```
_PAGE_RW → 13
#define _PAGE_RW       (_AT(pteval_t, 1) << _PAGE_BIT_RW)

_AT → /는 우리가 쓰는 라이브러리니까 아니고, 1 이나 2 로 간다
```

```
#ifdef __ASSEMBLY__ // 어셈블리가 설정되어 있으면 X
#define _AC(X,Y)        X
#define _AT(T,X)        X
#else // 설정되어 있지 않으면 ((T)(X)) T 타입의 X???
#define __AC(X,Y)       (X##Y)
#define _AC(X,Y)        __AC(X,Y)
#define _AT(T,X)        ((T)(X))
#endif // 어쨌든 X
```

즉_PAGE_RW 는 1 을 PAGE_BIT_RW 만큼 shift 한 값이다

나가서 _PAGE_BIT_RW → 2

```
#define _PAGE_BIT_RW 1
```

즉_PAGE_RW 는 1 을 PAGE_BIT_RW (2)만큼 shift 한 값이다

즉 , change_page_attr_set(&addr, numpages, __pgprot(_PAGE_RW), 0);

에서 __pgprot(2)

나가서 __pgprot(2) → 42

```
define __pgprot(x)     ((pgprot_t) { (x) } ) //  pgprot_t 의 x 타입? 매크로형태 2 가 들어갔지만 구조체 타입
// pgprot 들어가서 보면, pgprot 구조체에 2 를 넣었음을 알수있다
```

```
static inline int change_page_attr_set(unsigned long *addr, int numpages,
                                       pgprot_t mask, int array) //change_page_attr_set : 페이지 속성을 설정한다.
// 주소, 페이지 개수, 마스크는 2 였고, array 는 0 ← change_page_attr_set(&addr, numpages,
__pgprot(_PAGE_RW), 0);
{
        return change_page_attr_set_clr(addr, numpages, mask, __pgprot(0), 0,
                (array ? CPA_ARRAY : 0), NULL); // CPA_ARRAY 는 들어가면 2
// array 가 있으면 2 없으면 0 ← 인자 4 개 넘어왔는데 int array 는 0 으로 넘어왔음
}
```

```
//change_page_attr_set(addr, numpages=3??, 2, 0,0, NULL)
static int change_page_attr_set_clr(unsigned long *addr, int numpages,
                                    pgprot_t mask_set, pgprot_t mask_clr,
                                    int force_split, int in_flag,
                                    struct page **pages)
{
        struct cpa_data cpa;
        int ret, cache, checkalias;
        unsigned long baddr = 0;

        memset(&cpa, 0, sizeof(cpa)); // cpa 를 0 으로 memset
        /*
         * Check, if we are requested to change a not supported
         * feature:
         */
        mask_set = canon_pgprot(mask_set);  // mask_set 2
```

```
→ #define canon_pgprot(p) __pgprot(massage_pgprot(p))
// 페이지 property 값을 가지고 온다
→ static inline pgprotval_t massage_pgprot(pgprot_t pgprot)
        if (protval & _PAGE_PRESENT) // 2 랑 1 이랑 &니까 건너뛴다
                protval &= __supported_pte_mask;
return protval; // 2 를 가지고 온다, mask_set 2 은 여전히 2

        mask_clr = canon_pgprot(mask_clr); //mask_clr 는 0 → 똑같이 바뀌지 않음
        if (!pgprot_val(mask_set) && !pgprot_val(mask_clr) && !force_split)
// 건너뛴다, 쇼트컷 첫번째 인자에서 이미 결정남
                return 0;

        /* Ensure we are PAGE_SIZE aligned : 페이지 사이즈가 페이지사이즈 단위(시험때 2 의 n 승 단위 정
렬을 생각하자)로 정렬되었음을 보장 즉 &~할 것*/
        if (in_flag & CPA_ARRAY) { // 0 & 2 , 건너뛴다
                int i;
                for (i = 0; i < numpages; i++) {
                        if (addr[i] & ~PAGE_MASK) {
                                addr[i] &= PAGE_MASK;
                                WARN_ON_ONCE(1);
                        }
                }
        } else if (!(in_flag & CPA_PAGES_ARRAY)) { // 0&4 들어간다
                /*
                 * in_flag of CPA_PAGES_ARRAY implies it is aligned.
                 * No need to cehck in that case
                 */
  if (*addr & ~PAGE_MASK) {
//→ 30, PAGE_MASK       (~(PAGE_SIZE-1)) : 2 의 n 승 -1 에 ~연산
// 페이지사이즈 4096 나올것
// 마스킹 할게 있으면 값을 가지고 없으면 0
// 페이지 단위로 정렬이 되어있으면 마스킹 안된다
// 5000 & ~ 4096 = 4096, 9000 &~4096=8192

→ /* PAGE_SHIFT determines the page size */
#define PAGE_SHIFT      12
#define PAGE_SIZE       (_AC(1,UL) << PAGE_SHIFT) // 1<<12 는 4096
#define PAGE_MASK       (~(PAGE_SIZE-1))


                        *addr &= PAGE_MASK; // 즉, 4096 단위이거나 0 으로 셋팅
                        /*
                         * People should not be passing in unaligned addresses:
                         */
                        WARN_ON_ONCE(1);
→x86 은 없음
 #define WARN_ON_ONCE(condition) ({
        static bool __section(.data.unlikely) __warned;
        int __ret_warn_once = !!(condition);          // 이중부정, 그대로 → 컴파일러 최적화를 위함
```

```
            }
            /*
             * Save address for cache flush. *addr is modified in the call
             * to __change_page_attr_set_clr() below.
             */
            baddr = *addr; // 0 또는 4096 단위를 셋팅해준다
        }

        /* Must avoid aliasing mappings in the highmem code */
// highmem code 에 mapping 되는 것을 피한다. highmem 은 간접참조이므로 속도가 느리다 (시스템콜 0 번 table or vector table0x1000 이 간접참조에 있을리가 없다), x86 은 highmem 쓸일 없음
즉 32bit 전용코드

        kmap_flush_unused();// 하는게 없네
        vm_unmap_aliases(); //하는게 없다
// ARM 은 32bit – 32bit 라면 하는게 있겠지…?

        cpa.vaddr = addr;
        cpa.pages = pages; // NULL
        cpa.numpages = numpages; // 3
        cpa.mask_set = mask_set; // 2
        cpa.mask_clr = mask_clr; // 0
        cpa.flags = 0;
        cpa.curpage = 0;
        cpa.force_split = force_split; //0

        if (in_flag & (CPA_ARRAY | CPA_PAGES_ARRAY)) // 0 & 2 | 4 =6 → 0
            cpa.flags |= in_flag;
 /* No alias checking for _NX bit modifications */
        checkalias = (pgprot_val(mask_set) | pgprot_val(mask_clr)) != _PAGE_NX;
//  pgprot_val(2), _PAGE_Nx 와 같은가 → 2 의 63 승과 같을리가 없음
→ #define _PAGE_NX        (_AT(pteval_t, 1) << _PAGE_BIT_NX)  ← 2 의 63 승

        ret = __change_page_attr_set_clr(&cpa, checkalias); // cpa 를 가지고 __페이지 속성을 변경
→ static int __change_page_attr_set_clr(struct cpa_data *cpa, int checkalias)
{
        int ret, numpages = cpa->numpages;

        while (numpages) { // while(3)
            /*
             * Store the remaining nr of pages for the large page
             * preservation check.
             */
            cpa->numpages = numpages; // 이미 3 인데
            /* for array changes, we can't use large page */
            if (cpa->flags & (CPA_ARRAY | CPA_PAGES_ARRAY)) // 0 이겠네
                cpa->numpages = 1;

            if (!debug_pagealloc) // 커널컴파일시 셋팅해주어야 한다. 디버깅안할거니까 1
                spin_lock(&cpa_lock); // 여러 프로세스의 접근을 막는다 → 41min
```

```
        ret = __change_page_attr(cpa, checkalias);
        if (!debug_pagealloc)
            spin_unlock(&cpa_lock);
        if (ret)
            return ret;

        if (checkalias) {
            ret = cpa_process_alias(cpa);
            if (ret)
                return ret;


    /*
     * Check whether we really changed something:
     */
    if (!(cpa.flags & CPA_FLUSHTLB))
        goto out;

    /*
     * No need to flush, when we did not set any of the caching
     * attributes:
     */
    cache = !!pgprot2cachemode(mask_set);

    /*
     * On success we use CLFLUSH, when the CPU supports it to
     * avoid the WBINVD. If the CPU does not support it and in the
     * error case we fall back to cpa_flush_all (which uses
     * WBINVD):
     */
    if (!ret && cpu_has_clflush) {
        if (cpa.flags & (CPA_PAGES_ARRAY | CPA_ARRAY)) {
  cpa_flush_array(addr, numpages, cache,
                  cpa.flags, pages);
        } else
            cpa_flush_range(baddr, numpages, cache);
    } else
        cpa_flush_all(cache);

out:
 return ret;
}
```

cpa 로 들어가서???어디로 들어갔지??

cpa 가 뭔지 보면 current flusing context?? // context: 메모리 컨텍스트

pgd _pgd // 페이지 디렉토리 저장  22 분….??????

```
orig_call = (void *)sys_call_table[__NR_open];
    sys_call_table[__NR_open] = (void *)sys_our_open;
    write_cr0(cr0);
```

```
    printk("<0>Hooking Success!\n");
    return 0;
}
```