2000년 한국정보처리학회 추계 학술발표논문집 제 7 권 제 2 호

ARM 프로세서 기반 이동 단말기 플랫폼 설계

박관민*, 권오형*, 김도우*, 정민수*
*경남대학교 컴퓨터공학과
e-mail : drago@hawk.com.kyungnam.ac.kr

Design of mobile terminal platform based on ARM Processor

Kwan-min Park*,*, Oh-Hyung Kweon*, Do-woo Kim*,Min-soo Jung*
*Dept. of Computer Science, Kyung-nam University

요 약

현재 무선 인터넷 환경이 급속히 확산되고 있으며 그에 따른 모바일의 관심이 높아지고 있다. 모바일 단말기의 제약된 환경조건에서 효과적으로 수행할 수 있는 프로세서를 선택하고 프로세서의 특성을 부각시킬 수 있는 실시간 커널의 포팅을 통해서 모바일 환경에 적합한 플랫폼 구성을 중심으로 본 논문에서 서술하였다

1. 서 론

현재 IMT2000 에 따른 이동 통신을 통한 무선 인터넷 환경이 급격히 확산되고 그에 따른 이동 단말기의 기능 강화가 절대적으로 요구되고 있다.

기존의 단순한 문자 위주의 음성 통신으로 제한된 서비스에서 그래픽 위주의 화상을 통한 다양한 서비스와 인터넷과 연계가 구축되어가고 있다. 이러한 요구에 적당한 플랫폼을 위해서는 전력 소비가 낮고 높은 수행 능력을 지니고 제한된 환경에 적합한 프로세서의 선택과 프로세서와 이를 적절히 관리하고 응용 프로그램을 원활하게 운용할 운영 체제가 필요하다. ARM 프로세서는 일종의 축소 명령어 세트 컴퓨터 (RISC)의 일종으로 이동 단말기 환경의 제약 조건을 충분히 극복할 수 있도록 설계된 프로세서이다. 또한, RISC 의 기본적인 특징 외에 추가적인 특징으로 인해서 높은 수행 능력을 지니고 있으며 내장 가능하여 이식성도 뛰어나 이동 단말기 환경에 적합한 장점을 지니고 있다. 마이크로 C/OS-2 는 실시간 커널로 간단하고 명확한 소스가 제공되며 이식성이 뛰어나고 내장 가능하도록 설계 되었으며 프로세서의 플랫폼에 맞도록 불필요한 연산을 제거함으로써 크기가 축소 가능하다. 또한, 선점(preemptive)의 특징과 멀키테스킹(multitasking)이 가능하며 다양 한 서비스를 제공하며 신뢰성이 뛰어나다. 이러한 ARM 프로세서와 마이크로 C/OS-2 의 특징을 이용하여 mobile 환경에 적합한 플랫폼을 설계하고 다양한 응용 프로그램을 효과적으로 수행하고 관리한다.

본 논문의 구성은 다음과 같다. 2 장에서는 ARM 프로세서에 대한 기본적인 개요를 기술하고 3 장에서 마이크로 C/OS-2 에 대하여 기술한다. 4 장에서 ARM 을 기반으로 마이크로 C/OS-2 의 포팅(porting)에 대한 이동 단말기 플랫폼을 구성하고 5 장에서 결론 및 향후 과제를 제시한다.

2. ARM 프로세서

ARM 은 축소 명령어 세트 컴퓨터의 한 종류로 large uniform register file, load-store architecture, 단순한 주소 필드 모드(address mode) 그리고 일정하고 고정된 명령어 필드(field) 등의 일반적인 RISC 의 특징을 가지면서 추가적으로 산술 논리 장치(ALU)와 쉬프터(shifter)의 사용을 극대화하기 위해 산술논리장치와 쉬프트 둘 모두를 동시에 관리 가능하며 프로그램 루프(loop)를 최적화하기 위해

auto-increment 와 auto-decrement 주소 모드(addressing mode)를 가진다. 또한 데이터 처리율의 극대화를 위해서 다중 명령(multiple instruction)을 적재(load)하고 저장(store)하며 모든 명령을 조건적으로 수행하는 특징도 가지고 있다. 이러한 추가적인 특징으로 인해서 높은 수행 능력을 지니는 반면 낮은 전력 소비 그리고 작은 크기의 규격과 같은 제한된 환경에서 기본적인 RISC 구조보다 향상된 성능을 제공한다.

2.1 ARM의 레지스터

ARM은 30개의 범용 레지스터와 6개의 상태 레지스터(register) 그리고 하나의 프로그램 카운터를 가지며 특히, 주요한 16 개의 레지스터들은 비특권 코드(non-privileged code)로 다른 프로세서 모드로 전환이 불가능한 사용자 모드 레지스터로 제공된다. 레지스터 15는 프로그램 카운터(PC)로 사용되며 모든 ARM 명령은 32 비트의 하나의 워드(word)로 항상 워드 단위로 정렬되기 때문에 프로그램 카운터는 단지, 30 비트만을 가지며 하위 2 비트는 0 으로 채워진다. 레지스터 14 는 링크 레지스터(LK)로 사용되며 부프로그램 호출(subroutine call)이 요구되었을 때 사용된다. 14 개의 레지스터는 하드웨어가 아닌 소프트웨어에 의해서 정의된 것에 대해 사용된다. 특히, 레지스터 13은 스택 포인터로 이용된다.

2.2 예외 처리

ARM 은 fast 와 normal 의 두 레벨의 인터럽트와 메모리 abort, 정의되지 않은 명령의 수행과 소프트웨어 인터럽트의 5 가지 예외를 제공한다. 예외가 발생하면 일반적인 레지스터들은 예외 처리를 위한 예외 모드(exception mode)의 레지스터로 대체된다. 예외 처리 후에 링크 레지스터에 의해 예외 처리에 대한 반환 주소(return address)를 유지하고 있으며 링크 레지스터는 스택 포인터를 통해서 예외 처리자(exception handler)를 제공한다.

현재 수행중인 프로세서의 상태는 현재 상태 레지스터 CPSR(Current Program Status Register)에서 유지되는데 CPSR 은 Negative, Zero, Carry, Overflow 의 4 가지 condition code flag 를 가지며 2 개의 인터럽트 disable 비트와 현재 프로세서 모드를 인코드(encode)하는 5 비트로 구성된다. 예외 처리 모드에서는 SPSR(Saved Program Status Register)에서 예외 처리를 수행하기 전에 task 의 CPSR 을 유지하도록 한다. 예외가 발생했을 때 현재 명령의 실행은 중지되고 메모리의 고정된 위치에 존재하는 예외에 대한 벡터(exception vector)가 수행되는데, 운영 체제는 초기화될 때 모든 예외에 대한 처리자(handler)를 설치해야 하고 예외가 발생했을 때 상태에 대한 손실이 없는 예외 처리를 위해서 특권 연산 시스템 테스크(task)는 사용자 모드가 아닌 시스템 모드에서 수행하도록 해야 한다.

2.3 명령어 세트(Instruction Set)

ARM 프로세서는 branch, data-processing, load 와 store, coprocessor 4 가지 기본적인 명령어 클래스를 가진다. ARM 명령은 조건적으로 수행되어지며 data-processing instruction 은 처리 결과에 따라서 CPSR 의 4 개의 조건 코드 프레그들(condition code flags)을 변경할 수 있고 다음 명령은 condition code flag 상태에 따라 조건적으로 수행된다. 프로그램 카운터의 값을 변경에 의해서 제어 흐름을 바꾸는 분기 명령(branch instruction)을 가지며 이는 소프트웨어 인터럽트 SWI 를 발생하도록 할 수 있으며 이는 운영 체제에서 정의된 서비스를 요구하여 운영 체제를 호출 할 수 있다. Data-processing instruction 은 범용 레지스터에서 연산을 수행하는데 고유의 data-processing instruction, multiply instruction 과 status register transfer instruction 의 3 종류가 있다. 16 개의 산술 논리 명령의 가지고 있으며 2 개의 오프랜드(operand)를 통해 산출된 결과를 레지스터나 선택적이지만 condition code flag 에 저장되기도 한다. 그러나, data-processing instruction 은 그 결과 레지스터에 저장하지않고 operand 의 값을 비교하여 condition code flag 에 저장하는데 이러한 비교를 위한 오프랜드는 하나가 항상 레지스터를 나타내거나 직접적인 값 또는 선택적으로 쉬프트(shift)된 레지스터의 값이다. 하지만 ARM 은 shift 명령을 제공하지 않고 대신 프로그램 카운터의 값을 data-processing instruction 을 통해 직접 입력하여 shift 명령을 구현한다. 곱셈 명령은 32 비트 연산과 64 비트 연산의 2 가지의 종류가 있으며 이들은 연산 비트에 따라서 저장을 위해 하나 이상의 레지스터를 사용한다. 그리고, 상태 레지스터 전송 명령은 CPSR 또는 SPSR 의 내용을 범용 레지스터에 전송한다.

load 와 store 명령은 하나의 레지스터의 값을 load 또는 store 하거나 다중 레지스터의 값을 load 하고 store 하며 메모리 주소의 값으로 레지스터의 값을 전환하는 3 가지의 종류가 있다. coprocessor 명령은 data-processing instruction 에서 coprocessor 명시한 내부의 연산을 시작하고 레지스터 전송으로 ARM 레지스터로 부터 coprocessor 의 값을 전송하도록 하고 data-transfer instruction 으로 메모리에 coprocessor data 를 전송하거나 메모리로부터 전송 받는다.

3. 마이크로 C/OS-2 실시간 커널

마이크로 C/OS-2 는 주로 ANSI C 로 작성되고 프로세서를 명세하고 프로세서에 종속되는 일부 영역은 어셈블리어로 작성된 실시간 커널로 8 비트, 16 비트, 32 비트 뿐만 아니라 64 비트 마이크로프로세서를 지원할 수 있으며 내장 가능하도록 구성되어 있어 이식성이 뛰어나다. 또한, 마이크로 C/OS-2 의 전체에서 필요한 서비스만을 제공하기위한 분리가 용이하여 단말기의 특성에 따라서 운용 체제의 축소가 가능하여 운영 체제에 필요한 메모리의 양을 줄일 수 있다. 또한, 일반적인 상용 실시간 커널과 마찬가지로 마이크로 C/OS-2 는 완전한 선점 실시간 커널로서 항상 가장 높은 우선순위(priority)를 갖는 task 를 실행한다. 또한 multitasking 을 수행하는데 64 개 정도의 task 들을 관리할 수 있으며 시스템이 8 개를 예약하여 사용하고 사용자는 56 개의 task 를 수행할 수 있으며 각 task 는 유일한 우선순위(priority)를 할당받는다. 마이크로 C/OS-2 는 round-ronbin scheduling 을 지원하지는 않는다. 마이크로 C/OS-2 의 모든 함수와 서비스들의 실행 시간은 결정적이기 때문에 함수나 서비스의 실행시 소요되는 시간을 미리 알수 있다. 그리고, 각 task 는 자신의 스택을 가지며 각 스택의 크기는 일정한 것이 아니라 task 의 특성에 따라서 서로 다르다. 따라서, 응용 프로그램에 필요한 RAM 의 크기가 감소시킬 수 있고 스택의 크기를 검사하는 특징을 지니고 있어 task 가 필요로 하는 스택의 크기를 결정할 수 있도록 해 준다. 또한 메일 박스, 큐, 세마포어, 고정된 크기의 메모리 영역과 시간 관련한 기능 등 다양한 서비스를 제공하며 인터럽트를 255 단계로 구분하여 인터럽트를 관리한다. 따라서, 마이크로 C/OS-2 는 이식이 용이하고 ROM 으로 내장가능하며 크기를 최소화 할 수 있어 메모리를 관리가 쉽도록 설계되어 제약된 환경을 제공하는 이동 단말기 환경에 적합한 실시간 커널이라 할 수 있다.

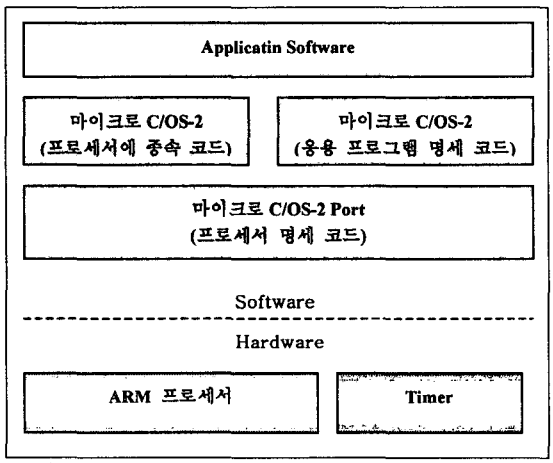
4. ARM 기반의 플랫폼 설계

4.1 마이크로 C/OS-2 의 제약 조건

마이크로 C/OS-2 를 수행하기 위해서는 프로세서가 기본적으로 갖추어야 할 조건이 몇 가지 따른다. 먼저 프로세서가 재진입 코드(reentrant code)를 생성할 수 있어야 하며 인터럽트를 제공해야 한다. 또한 인터럽트의 enable 과 disable 이 가능해야 한다. 그리고 하드웨어적인 스택이 제공되어야 하며 메모리나 스택 그리고 CPU 레지스터, 스택 포인터를 load 하고 store 하는 명령을 가져야 한다. ARM 프로세서는 기본적으로 ANSI C 와 어셈블리어를 지원할 뿐만 아니라 객체 지향의 C++까지도 지원 가능한 광범위한 이식성을 지니고 있으며 이를 통해서 재진입 코드의 생성이 가능하다. 그리고 ARM 프로세서는 CPSR 에 인터럽트 disable 비트를 가지고 있어 CPSR 의 상태 비트 값을 통해 인터럽트의 enable 과 disable 가능하다. 그리고 ARM 프로세서의 레지스터 13 은 스택 포인터로서 스택뿐만 아니라 CPU 레지스터에도 load 및 store 가 가능하다. 따라서 ARM 프로세서는 마이크로 C/OS-2 의 제약 조건들을 충분히 만족시킨다.

4.2 ARM 기반 플랫폼의 구성

플랫폼의 구성을 위한 기본적인 구조는 그림 1 에서 보는 바와 같이 하드 웨어와 관련된 부분은 크게 포팅에 관련한 영역과 프로세서에 종속되는 영역으로 구분한다.



[그림 1] 마이크로 C/OS-2 와 ARM process 의 구조

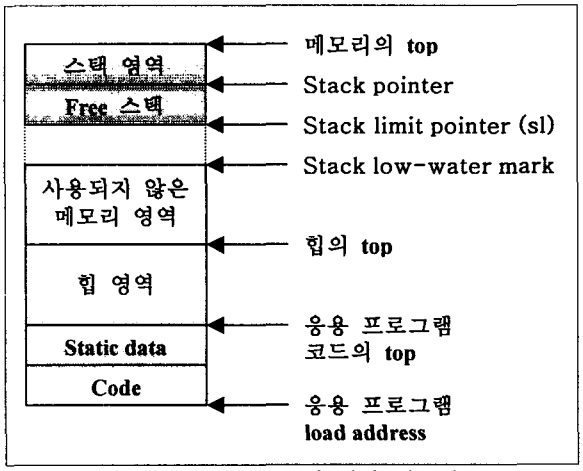
포팅에 관련한 영역 즉, 프로세서 명세 코드 부분에서는 프로세서와 관련한 전역 변수들에 대해서 설정하고 데이터 타입을 선언하고 매크로를 정의한다. 그리고 하드웨어적으로 제공될 스택에 대한 함수를 명시해 주고 프로세서와 관련된 명세들과 인터럽트 그리고 문맥 전환등을 정의한다.

데이터의 타입을 선언에 있어 ARM 프로세서의 워드의 길이에 알맞도록 데이터 타입을 정의하고 define 문을 사용하여 매크로를 정의한다. 스택을 사용하기 위해서 스택 포인터와 크기에 대한 초기화가 필요하며 스택의 최저 상태를 지정하여야 한다. 그리고, 최저 상태를 벗어난 접근에 대해서 low-level failure 명시하고 그에 따른 연산을 제공한다.

ARM 프로세서는 user, FIQ, IRQ, supervisor, abort, undefined, system 의 7 가지 프로세서 모드를 지원하며 그에 따른 예외 처리를 위한 인터럽트가 발생하고 이 인터럽트 수행을 위한 인터럽트 벡터를

가진다. 그리고 프로세서 명세 코드에서 이러한 인터럽트에 대하여 정의하는데 소프트웨어 인터럽트 서비스 루틴에 대한 설치와 IRQ 에 대한 enable 및 disalble 함수 그리고 IRQ 인터럽트에 대한 재설정한다. 또한, 소프트웨어 인터럽트(SWI)에 대한 설치와 연산을 제공한다. 현재 프로세서 상태 레지스터를 변경하는 FIQ 와 IRQ 를 disable 하여 인터럽트 disable 을 하도록 하고 인터럽트를 enable 하면 FIQ 와 IRQ 모두 enable 되고 현재 상태 레지스터 CPSR 의 I 와 F 비트의 값이 reset 되면서 이전의 값으로 다시 돌아올 수 있다. 인터럽트의 정의함으로서 임계 영역의 접근할 수 있도록 하고 상태 레지스터의 값을 통한 문맥 전환 (context switch)을 제공한다.

그림 2 는 프로세서 명세 코드에 따른 메모리 맵의 기본적인 구조를 보여준다. ARM 은 메모리 맵을 구성하고 그에 대한 이미지를 통해서 ROM 화하고 내장할 수 있다.



[그림 2] 프로세서 명세 코드에 대한 메모리 맵

메모리 맵은 크게 코드 영역과 데이터 영역으로 구분되며 코드 영역의 값은 변화되지 않는 코드들로 구성되며 변수값과 같이 변경되는 값에 대해서는 데이터 영역에서 작성된다. 그리고 코드 영역을 명령은 데이터 영역의 주소값에 대한 포인터를 이용하여 데이터 영역의 값에 접근할 수 있다.

메모리 맵의 CODE 영역에서 스택 포인터를 초기화하고 스택 크기에 대한 초기화, 최저 상태를 지정 등과 같은 스택에 관련한 사항과 힙의 관리, 그리고 low-level failure 에 연산과 마이크로 C/OS-2 의 stack check 를 지원에 대하여 명시하며 일반적인 ARM 프로세서 함수 처리에 대하여 함수를 작성한다. 그리고 여기서 마이크로 C/OS-2 의 메인(main)을 호출한다. 즉 프로세서에 종속되는 코드를 호출하게 된다.

프로세서 종속의 코드 영역은 마이크로 C/OS-2 의 핵심으로 실질적인 마이크로 C/OS-2 의 수행에 대하여 명시하는 영역이라고 할 수 있다.

마이크로 C/OS-2 에서 사용할 전역 변수 및 지역 변수를 선언하고 우선순위 할당을 위한 테이블을 정의한다. 그리고, 마이크로 C/OS-2 의 내부 데이터 구조와 ARM 프로세서를 초기화하고 스케줄러와 멀티 테스킹을 시작하는 함수를 제공한다. 또한, 스케줄러의 lock 과 unlock 함수를 통해서 자원에 대한 선점을 제어하도록 해 준다. 또한 task 의 생성과 삭제, task 우선 순위의 변경을 위한 함수를 제공한다. 마이크로 C/OS-2 의 시스템에 의해 수행될 idle task 에 대해 명시하고 시스템의 timer 와 관련한 tick 처리를 수행한다. 세마포어에 대한 생성과 대기, 그리고 세마포어에 대해서 알리는 함수를 작성하고 task 간의 통신을 위해서 메일 박스를 생성하고 메일 박스로 메시지를 보내고 메일 박스로부터 메시지를 대기하는 함수와 큐를 생성하여 메시지를 보내고 대기하는 함수를 작성한다.

5. 결론 및 향후 과제

본 논문에서 제시한 ARM 기반의 모바일 단말기 플랫폼은 마이크로 C/OS-2 를 ARM 프로세서에 맞도록 변환함으로써 ARM 프로세서의 효과적인 수행을 도모한다. 이를 기반으로 무선 인터넷 환경에서 수행될 응용 프로그램이 요구된다. 특히, 자바 환경을 통한 응용 프로그램의 개발이 활발히 수행되고 있고 더욱이 KVM 을 이용한 응용 프로그램에 대한 관심이 높다. 따라서 이 연구를 바탕으로 KVM 을 이용한 응용 프로그램과의 연계와 그를 통한 이동 단말기 플랫폼의 구현에 대한 연구가 필요하다.

참고 문헌

[1] MicroC/OS-II The Real-Time Kernel, Jean J. Larosse
[2] ARM Software Development Toolkit User Guide
[3] ARM Software Development Toolkit Reference Guide
[4] ARM architecture Reference Manual, Jagger
[5] ARM Target Development System User Guide
[6] ARM Firmware Suite Reference Guide
[7] http://www.arm.com/