저궤도 위성용 탑재소프트웨어 개발을 위한 ERC32 프로세서 소개

이재승*, 최종욱*, 채동석*, 이종인*, 김학정*
*한국항공우주연구원
e-mail : jslee@kari.re.kr

An Introduction to ERC32 to Develop Flight Software for LEO Satellites

Jae-Seung Lee*, Jong-Wook Choi*, Dong-Seok Chae*, Jong-In Lee*, Hak-Jung Kim*
*Korea Aerospace Research Institute

요 약

유럽에서는 위성에 탑재할 고성능 탑재컴퓨터로 MCM-ERC32 보드를 개발하여 사용하고 있다. 이에 한국항공우주연구원에서는 향후 개발되는 저궤도 관측위성에 사용할 고성능 탑재컴퓨터로 MCM-ERC32 를 적용할 예정이다. 현재까지 한국항공우주연구원에서 개발된 저궤도 관측위성은 Intel 계열의 CPU 를 탑재한 컴퓨터를 사용하였으며, MCM-ERC32 에 대한 개발기술은 전무한 상태이다. 따라서, MCM-ERC32 로의 탑재컴퓨터 변경은 전체적인 시스템의 재설계가 요구되며, 이를 이용한 탑재소프트웨어의 개발에도 많은 영향을 미치게 된다.

본 논문에서는 MCM-ERC32 를 이용한 새로운 탑재컴퓨터 시스템에 적용 가능한 탑재소프트웨어 개발을 위해 ERC32 프로세서의 Integer Unit 의 고유한 기능에 대해 소개한다.

1. 서론

유럽에서 차세대 위성의 개발을 위한 고성능 탑재컴퓨터로 사용하기 위하여 MCM-ERC32 를 개발하였다. MCM-ERC32 는 ESA(European Space Agency)의 지원 하에 개발되었으며, 크게 다음과 같은 4 부분으로 구성되어 있다.

- ERC32 Single Chip
- ASIC VASI(Very Advanced Sparc Interface)
- 6 MBytes SRAM (protected by EDAC)
- 32 MBytes DRAM (protected by Reed-Solomon)

MCM-ERC32 는 위의 구성요소들이 위성용으로 사용 가능한 부품들을 사용하여 하나의 보드에 집적하여 제작된 보드이다. 향후 개발될 저궤도 관측위성의 탑재컴퓨터로 MCM-ERC32 가 사용될 예정이며, 이에 따라 MCM-ERC32 에서 실행 가능한 탑재소프트웨어의 개발이 진행되고 있다. 현재 이러한 탑재소프트웨어 개발의 시작단계로서 탑재컴퓨터의 기능 분석 및 BSP (Board Support Package) 설계가 수행되고 있으며, 특히 탑재컴퓨터의 프로세서 분석에 대한 중점적인 연구가 이루어지고 있다. ERC32(Embedded Real-time Computer 32-bit) 프로세서는 MCM(Multi-Chip Module)의 핵심부분으로 현재까지 저궤도 관측위성에 사용된 Intel 계열의 프로세서와는 기본 개념에서부터 다른 아키텍처를 가지고 있으며 내장형 프로세서의 다양한 기능을 가지고 있다.

본 논문에서는 MCM-ERC32 를 이용한 새로운 탑재컴퓨터 시스템에 적용 가능한 탑재소프트웨어 개발을 위해 ERC32 프로세서의 구성요소 중 Integer Unit 에서 제공되는 고유한 기능에 대해 설명한다. 프로세서의 기능에 대한 소개는 그 범위가 방대하므로 기존의 Intel 계열과 구별되는 ERC32 가 가지는 핵심적인 기능들을 중점적으로 소개한다. 특히 ERC32 는 유럽에서 자체적으로 개발되어 사용되는 프로세서이므로 이와 관련된 기술자료나 설계경험 등이 우리나라에서는 현재 전무한 상태이다. 본 논문에서 소개하는 ERC32 프로세서에 대한 분석 내용은 한국항공우주연구원에서 향후 위성탑재컴퓨터 및 탑재소프트웨어의 개발을

위해 수행된 MCM-ERC32 에 대한 기술분석을 통해 이루어졌으며, 다음의 그림 1 은 본 논문에서 소개하는 MCM-ERC32 보드의 전체적인 구성도를 보여준다.

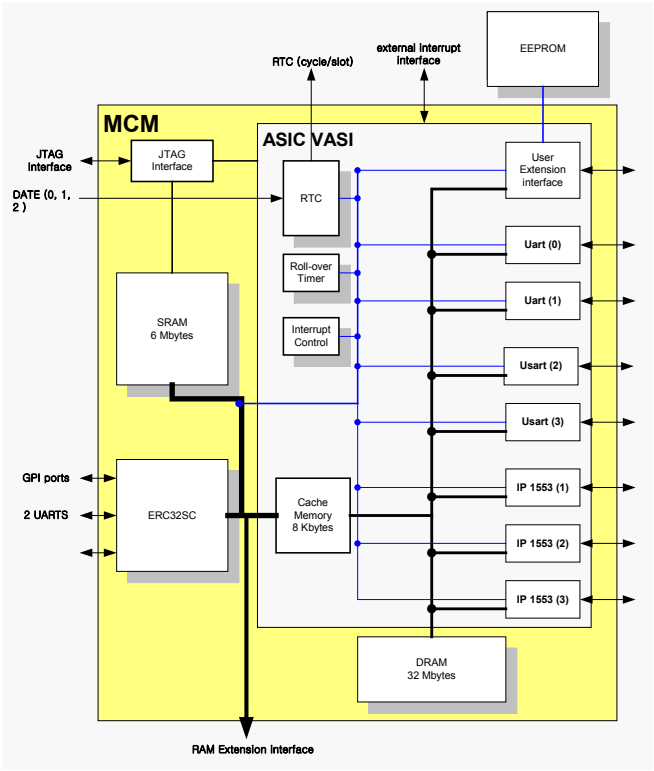

그림 1. Block Diagram of MCM-ERC32

2. ERC32 의 구성

ERC32 라고 불려지는 Rad Hard 32-bit SPARC Embedded Processor(TSC695F, [1])는 ESA 에서 개발된 프로세서로서 SPARC 아키텍쳐 V7[2,3]을 기본으로 하는 고집적, 고효율성을 가진 32 비트 RISC(Reduced Instruction Set Computer) 내장형 프로세서이다. ERC32 는 정수연산을 위한 IU(Integer Unit, TSC691E, [4]), 실수연산을 위한 FPU(Floating-Point Unit, TSC692E), 메모리 제어를 위한 MEC(Memory Controller, TSC693E)와 DMA Arbiter 를 내장하고 있다. 다음의 그림 2 는 ERC32 의 구성도를 나타낸다.

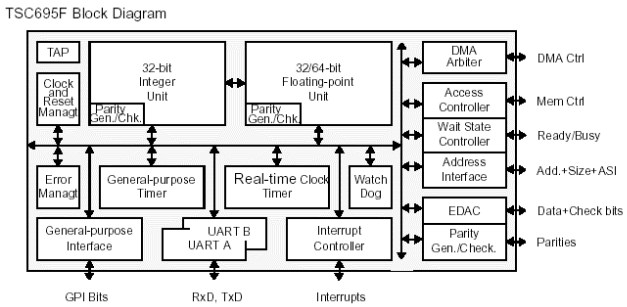

그림 2. Block Diagram of TSC695F

실시간 응용 프로그램을 위하여 TSC695F 는 워치독 타이머와 2 개의 범용 타이머, 인터럽트 컨트롤러, 병렬/직렬 인터페이스를 제공한다. 또한 오류 정정을 위하여 내/외부 버스에 대한 parity 를 제공하며, 외부 데이터 버스에 대한 EDAC 기능을 지원한다. 테스트 환경을 위해서는 OCD(On-Chip Debugger)와 JTAG 인터페이스를 제공하고 있다.

ERC32 의 RISC 칩은 Intel 계열의 CISC(Complex Instruction Set Computer) 칩과 달리 다음과 같은 특징을 가지고 있다.

- Simple Instruction Set : RISC 의 경우 명령어는 간단하고 기본이 되는 명령어로만 구성이 되어 있으며, 이런 간단한 명령어의 조합을 통해서 복잡한 명령을 수행할 수 있다.
- Same Length Instructions : 모든 명령어는 동일한 크기를 가지고 있어서 한번의 fetch 를 통해서 수행될 수 있다.
- Reduced Memory Access : load/store 명령만이 메모리에 접근할 수 있다. Load/store 명령어는 메모리와 레지스터 사이에 동작을 하게 되며, 이것은 하드웨어 컨트롤을 간략하게 하며 사이클 타임을 최소화할 수 있다.
- Small Number of Addressing Mode : 메모리에 접근할 때 short displacement, long displacement, indexed mode 의 3 가지만을 지원한다.
- 1 Machine-cycle Instruction : 대부분의 명령어는 1 번의 사이클에 완료되며 이 기능은 동일 시간에 여러 개의 명령어를 수행할 수 있게 해준다. RISC 의 속도를 올리는 pipeline 의 가장 핵심이 되는 기술이다.
- Pipelining : 한번에 한 개 이상의 명령어를 수행할 수 있게 해주는 기술로 다음 명령어를 수행하기 전에 이전 명령어가 완료되기를 기다릴 필요가 없다. SPARC V7 에서는 fetch, decode, execute, write 의 4 개의 단계를 가지게 되며 병렬적으로 각 단계는 수행된다.

ERC32 에서 제공하는 레지스터 모델을 그림 3 에 나타내었다.

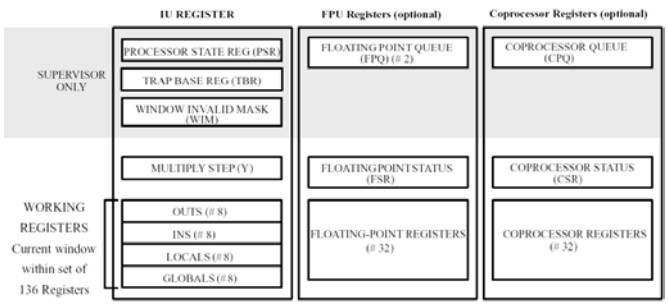

그림 3. TSC695F Register Model

IU 는 PSR(Processor Status Reg.), TBR(Trap Base Reg.), WIM (Window Invalid Mask), Y(Multiply Reg.), 8 개의 출

력 레지스터, 8 개의 입력 레지스터, 8 개의 로컬 레지스터, 8 개의 전역 레지스터를 제공하고 있다. FPU 레지스터는 수치연산에 필요한 레지스터로서 3 개의 FPQ (Floating-point Queue)와 FSR(Floating-point Status Reg.), CSR(Coprocessor Status Reg.), 32 개의 코프로세서 레지스터를 제공하지만 현재 ERC32 에서는 코프로세서를 지원하지 않고 있다.

3. ERC32 Integer Unit (TSC691E)

IU 는 프로세서의 전반적인 동작을 제어하며 대표적인 역할은 다음과 같다.

- 정수연산 명령 수행
- load/store 를 위한 메모리 연산 수행
- PC(Program Counter) 관리
- FPU 와 CP 수행을 위한 명령 관리

IU 에서 적용되는 pipeline 은 다음과 같은 4 단계로 이루어진다.

- Fetch : 명령어를 가져오기 위한 주소를 생성
- Decode : Instruction 레지스터에 명령어를 저장하고 해석, 프로세서는 레지스터에서 operand 를 읽고 다음 명령어의 주소를 계산
- Execute : 명령어를 수행하고 임시 레지스터에 결과를 저장
- Write : 결과를 해당 레지스터에 저장

위의 4 단계는 병렬적으로 수행되며, 4 개의 다른 명령이 동시에 수행되는 방식을 가지게 된다. 다음의 그림 4 는 IU 의 pipelining 에 대한 개념을 보여준다.

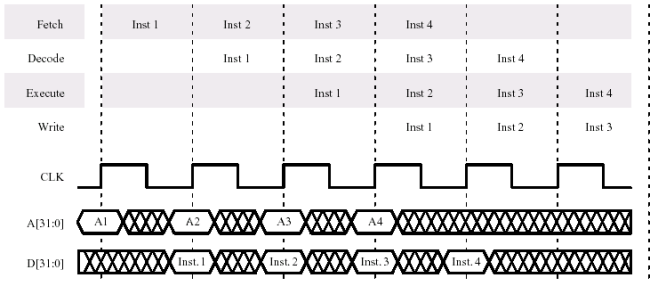

그림 4. Pipeline with Single-cycle Instructions

ERC32 의 IU 는 32-비트의 제어 및 상태 레지스터와 136 개의 범용 레지스터를 가지고 있으며, 각 레지스터들의 기능 및 특징에 대해서는 참고문헌을 통하여 자세하게 확인할 수 있다. 본 논문에서는 ERC32 에서 제공하는 윈도우 방식의 레지스터 파일 기능을 중심으로 설명한다.

4. Window Registers

본 절에서는 MCM-ERC32 에서 제공하는 레지스터 제어방식인 IU 윈도우 레지스터의 기능에 대해 설명한다.

ERC32 IU 의 136 개 범용 레지스터는 8 개의 전역 레지스터와 128 개의 윈도우 레지스터로 나뉘어지며, 128 개의 윈도우 레지스터는 다시 8 개의 그룹으로 나누어지며, 각 그룹은 ins, locals, outs 레지스터로 구성된다. 이러한 개념을 통해 윈도우 레지스터가 생성된다.

SPARC 은 최대 32 개의 윈도우가 제공 가능하며 현재 ERC32 에서는 8 개의 윈도우를 제공하고 있다. ERC32 의 윈도우 개념에서는 32 개의 active 레지스터 (24 개의 윈도우 레지스터와 8 개의 전역 레지스터)만을 억세스할 수 있다. 표 1 은 ERC32 윈도우 레지스터의 어드레싱을 보여준다.

표 1. Register Addressing

Register Number	Name
r[24] ~ r[31]	ins
r[16] ~ r[23]	locals
r[8] ~ r[15]	outs
r[0] ~ r[7]	globals

ERC32 에서 레지스터 파일은 원형 스택 개념으로 구현되어 있으며, 그림 5 처럼 윈도우 #7 에서 윈도우 #0 까지 8 개의 윈도우로 구성되어 있다.

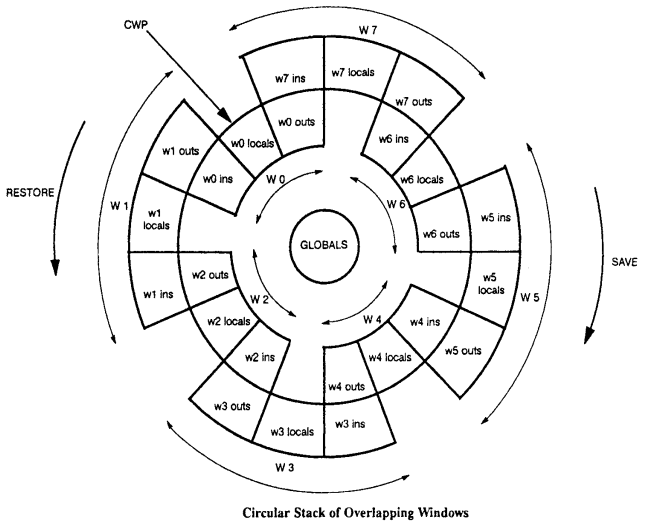

그림 5. Circular Stack of Overlapping Windows

앞에서 설명한 것처럼 한 개의 윈도우는 24 개의 윈도우 레지스터와 8 개의 전역 레지스터로 구성되어지며, 각 윈도우는 인접한 윈도우와 ins, outs 레지스터를 공유한다. 다음의 그림 6 에서와 같이 CWP(Current Window Pointer)+1 에 있는 outs 레지스터는 CWP 의 ins 레지스터가 되고, CWP 의 outs 레지스터는 CWP-1 의 ins 레지스터가 되는 방식을 이용한다. 인접한 윈도우의 ins 와 outs, globals 레지스터가 공유되는 반면 locals 레지스터는 오직 해당 윈도우에서만 사용된다.

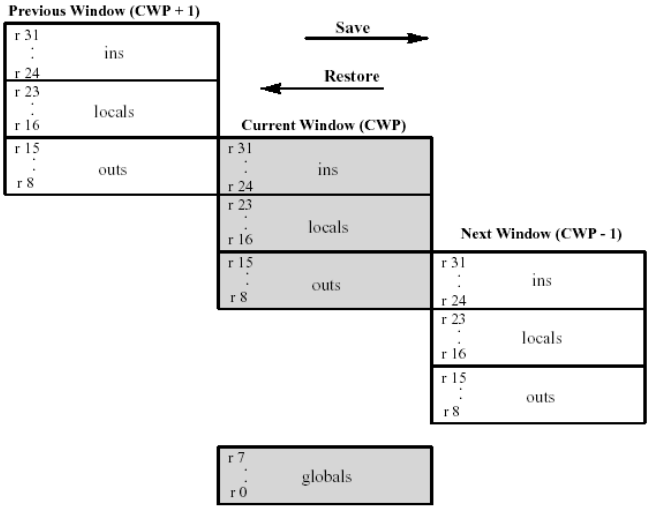

그림 6. Overlapping Windows

ERC32 의 레지스터 윈도우 overlap 은 프로시저의 호출과 리턴 시 파라미터들을 효율적으로 전송하는 기능을 제공한다. 만약 outs 및 globals 레지스터로 보낼 수 있는 파라미터들 보다 더 많은 파라미터를 전송해야 한다면 메모리 스택을 전송하는 방식을 이용한다. 윈도우 레지스터의 outs 레지스터 #6 은 스택 포인터(SP, Stack Pointer)로 메모리 스택의 주소로 사용되며, 스택 포인터는 호출된 프로시저에서 ins 레지스터 #6 이 되어 프레임 포인터(FP, Frame Pointer)라는 이름으로 사용된다.

이상에서 설명한 윈도우 레지스터의 기능을 테스트하기 위하여 Gaisler Research Lab 에서 개발된 LECCS 컴파일러와 ERC32 시뮬레이터인 TSIM Pro 를 이용하였다. 그림 7~8 에 태스크 스위칭에 따른 윈도우 레지스터의 기능에 대한 테스트 결과를 나타내었다.

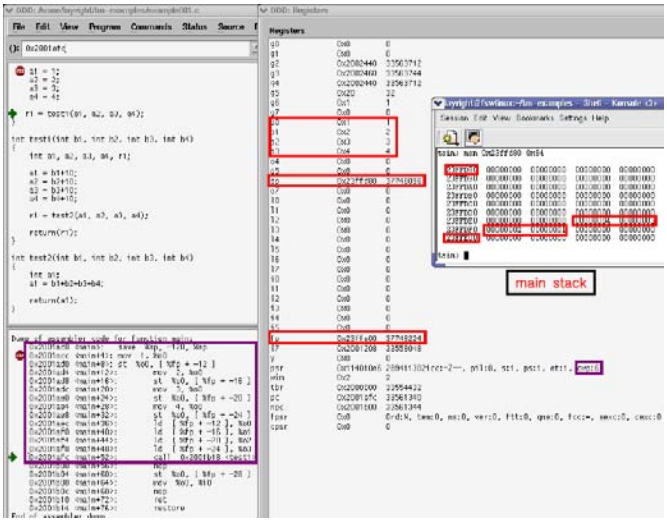

그림 7. Test for Window Registers (Step #1)

그림 7 은 test1 함수를 호출하기 전의 레지스터 상태를 보여준다. o0, o1, o2, o3 레지스터 값이 각각 1, 2, 3, 4 이며 SP 와 FP 사이의 영역을 덤프하여 1, 2, 3, 4 값이 존재하는 것을 확인할 수 있다. 'main' 함수가 사용하는 내부 스택 영역은 SP(0x23ffd80) ~ FP(0x23ffe00)인 것을 확인할 수 있다.

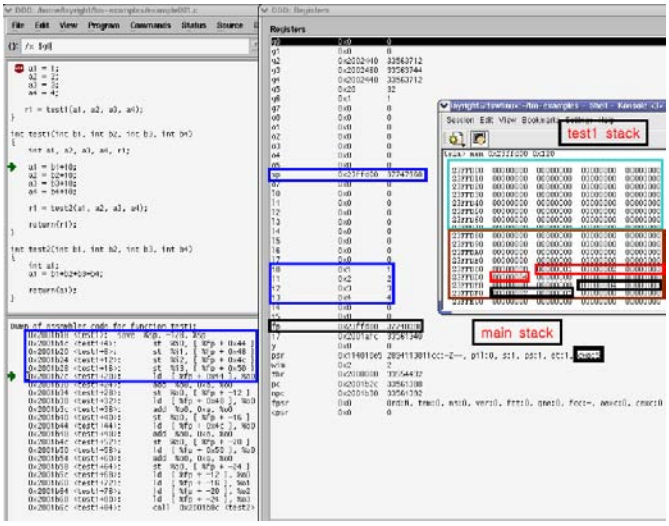

그림 8. Test for Window Registers (Step #2)

그림 8 은 main 에서 test1 을 호출하였을 때 레지스터의 상태를 보여준다. i0, i1, i2, i3 의 값이 1, 2, 3, 4 로 바뀐 것을 볼 수 있으며, 이것은 main 의 outs 레지스터가 test1 의 ins 레지스터가 되었다는 것을 말한다. 또한 CWP 가 6 에서 5 로 바뀌었으며, 각각 0x23ffd00 과 0x23ffd80 으로 바뀐 SP 와 FP 값을 통하여 test1 이 사용하는 스택 영역을 알 수 있다.

5. 결론

본 논문에서는 향후 저궤도 관측위성 개발에 사용할 고성능 탑재컴퓨터인 MCM-ERC32 용 탑재소프트웨어를 개발하기 위한 준비단계로서 ERC32 프로세서에 대한 주요 기능에 대하여 소개하였다. 추후 MCM-ERC32 에 대한 분석을 완료하면 최소 기능의 BSP 설계 및 개발과 함께 실시간 운영체제와의 연동 개발을 위한 프로세서의 정확한 메커니즘 분석과 이에 대한 테스트가 수행될 예정이다.

참고문헌

[1] "TSC695F SPARC 32-bit Space Processor User Manual", Atmel, 2003
[2] "SPARC V7.0 Instruction Set for Embedded Real-time 32-bit Computer(ERC32) for SPACE Applications", Atmel, 1996
[3] "The SPARC Architecture Manual", SPARC International Inc., 1992
[4] "TSC691E Integer Unit User's Manual", Atmel, 1996