다목적실용위성 2호 코프로세서를 위한 수치연산프로그램 설계

최종욱[0] 천이진 이재승
한국항공우주연구원
{jwchoi, yjcheon, jslee}@kari.re.kr

Design of Numerical Functions for KOMPSAT-2 Coprocessor

Jong-Wook Choi[0] Yee-Jin Cheon Jae-Seung Lee
Space Division, Korea Aerospace Research Institute

요 약

다목적실용위성 2호(KOMPSAT-2)에서는 위성자세 제어를 담당하고 있는 RDU(Remote Drive Unit)의 성능 향상을 위하여 80386 CPU와 함께 80387 Coprocessor를 장착하여 임무수행을 담당한다. 다목적실용위성 1호(KOMPSAT-1)에서는 수치연산을 위하여 상용소프트웨어인 PACLIB를 사용하여 수치연산을 80C186을 이용한 에뮬레이션 방식으로 수행하였지만, 2호기에서는 실질적인 코프로세서를 이용한 수치연산을 수행하게 된다. 본 논문에서는 다목적실용위성 2호에서 사용되는 80387 코프로세서의 초기화 과정과 예외사항 발생시 처리 방법, 80387 코프로세서를 이용한 수치연산 함수 구현 및 library 구성 방법에 대하여 설명한다.

1. 서 론

현재 개발중인 다목적실용위성 2호에서는 성능 향상을 위하여 다목적실용위성 1호와 달리 CPU가 80C186에서 80386으로 변경되었으며, 원격구동장치(RDU, Remote Drive Unit) 경우 수치연산을 위하여 인텔 80387 코프로세서가 장착되었다. 다목적실용위성 1호에서는 수치연산을 위하여 상용소프트웨어인 'PACLIB Floating Point Library'를 사용하여 수치연산이 80C186 CPU에 의한 에뮬레이션 방식으로 이루어졌다. 다목적실용위성 2호의 RDU의 경우 Intel 80387 코프로세서를 이용한 수치연산을 수행하기 위해서는 80387에서 지원하는 수치연산 명령어를 이용하는 수치연산 프로그램이 구현되어야하며, 80387 코프로세서의 초기화 및 FPU(Floating Point Unit) ISR(Interrupt Service Routine)에 대한 처리도 반드시 구현되어야 한다. 현재 다목적실용위성 2호 탑재소프트웨어 개발시 사용하는 Microsoft Visual C++1.52에서 제공하는 수치연산 함수는 자체 library형태로 지원되며, DOS를 기반으로 하고 있기 때문에 경성 실시간 시스템인 다목적실용위성 2호에서는 사용이 제한되며, 80187 코드만을 제공하기 때문에 80387 고유 명령어를 지원하지 않는 문제, 또한 library형태로 제공되어 소스를 지원하지 않는 등 많은 문제점을 가지고 있다.

본 논문에서는 다목적실용위성 2호에서 사용되는 80387 코프로세서의 초기화 과정과 예외사항 발생시 처리 방법, 80387 코프로세서를 이용한 수치연산 함수 구현 및 library 구성 방법에 대하여 설명한다.

2. Intel 80387 Coprocessor 구조

인텔 80387 코프로세서는 8개의 80bit 수치계산용 register stack, 8개 register stack의 상태를 알려주는 Tag Field, FPU의 상태를 알려주는 status register, FPU의 옵션을 조정하는 control register 등으로 구성되어 있다. 그림 1은 80387 코프로세서의 레지스터들 보여준다.

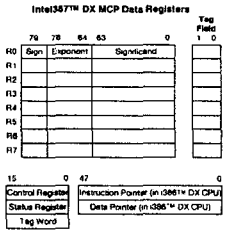

그림 1. 80387 Coprocessor Registers

그림 2에 나타나 있는 control register는 FPU의 연산시 발생하는 exception을 control 할 수 있도록 해주며, precision과 rounding 방법을 선택할 수 있도록 해준다. 80387를 이용한 연산시 발생하는 exception은 Precision, Underflow, Overflow, Zero Divide, Denormalized Operand, Invalid Operation 총 6가지이며, control register의 exception mask bit의 세팅유무에 따라, FPU가 exception을 처리할 것인지, ISR루틴을 통해 처리할 것인지를 결정하게 된다. 만약 해당 exception이 mask되어 있을 경우에는 FPU가 자동적으로 정해진 룰에 의해서 exception을 처리하게 되고, 계속 다른 명령을 수행하게 된다. 80387 status register는 FPU의 전반적인 상태를 나타내는 register이다. 6개의 exception의 발생유무와, FPU stack register의 상태, FPU 연산결과의 따른 flag 정보를 가지고 있다.

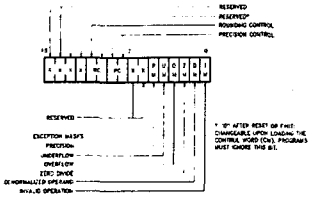

그림 2. 80387 Coprocessor control register

2.1 80387 코프로세서 초기화

시스템이 부팅 과정중에 80386 CPU가 초기화 되면서 80386은 코프로세서의 존재를 확인하게 된다. 확인 결과를 CR0의 ET bit에 저장하게 되고, FINIT 명령어를 이용하여 코프로세서를 초기화하게 된다. 코프로세서가 초기화 되면 FPU control register는 초기값 0x037F로 설정되며, 다목적실용위성 2호에서 사용될 control register로 재설정하게 된다. 다목적실용위성 2호에서 사용될 FPU의 control 방식은 precision exception만 mask가 되어 FPU 자체적으로 exception을 처리하게 하였으며, 나머지 5가지의 exception에 대해서는 FPU ISR루틴에서 처리하도록 하였다. Precision 경우는 Extended Precision 방식인 64bit를 지원하도록 하였으며, rounding 경우에는 chop방식을 지원하도록 하였다. 그림 3은 FPU 초기화 및 control register의 설정이후의 FPU의 상태를 보여준다.

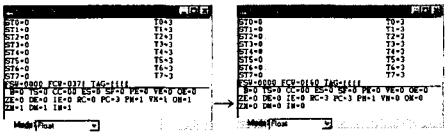

그림 3. 80387 Coprocessor 초기화

2.2 80387 코프로세서 Exception을 위한 FPU ISR

80387은 FPU status register의 6개의 exception 상태를 저장하고 있으며, control register의 exception bit의 setting 유무에 따라 FPU 또는 ISR루틴에서 처리하게 된다. 80387과 관련된 인터럽트는 7번(Coprocessor Not Available), 9번(Coprocessor Segment Over run), 16번(Coprocessor Error)이다. 인터럽트 7번인 경우는 코프로세서가 존재하지 않을 경우 발생하며 이는 코프로세서를 초기화 하는 과정에서 확인되며, 9번인 경우는 오직 보호모드(protected-mode)에서만 발생하므로 real-mode를 사용하는 다목적실용위성 2호 운영시스템에서는 Invalid ISR로 처리하게 된다.

FPU에서 mask되지 않은 exception 5가지 대해서는 인터럽트 16번이 발생하게 되며, floating-point exception handler를 호출 하게 된다. 다목적실용위성 2호에서는 다음 그림 4와 같은 단계를 거쳐 FPU exception을 처리하게 된다. 그리고 지상에서 FPU가 발생한 시간과 횟수, 그리고 발생한 위치와 그때의 FPU status register 정보를 확인할 수 있도록 모든 에러가 로그되게 된다.

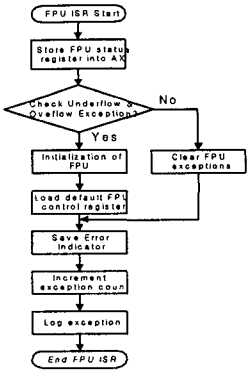

그림 4. 80387 Coprocessor Interrupt Service Routine

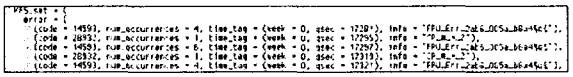

그림 5. 80387 코프로세서 에러 Log

FPU의 ISR처리에도 불구하고 위성시스템에 큰 영향을 미칠 경우에는 FM(Fault Management)에서 처리하여 위성의 안전을 담당하게 된다.

3. 80387 코프로세서를 이용한 수치연산 함수

다목적실용위성 2호의 RDU에서 사용하고 수치연산 함수로서는 sin, cos, tan, asin, acos, atan, atan2, fabs, floor, sqrt, fmod로서 대부분 위성 자세제어를 위해서 필요로 하는 함수들이다. 이상의 함수들은 MASM 6.11을 이용하여 어셈블러로 구현되어 졌으며, 위성탑재 소프트웨어 build시 library형태로 지원이 된다.

3.1. 80387 코프로세서를 이용한 삼각함수 구현

삼각함수는 80비트로 이루어진 8개의 83087 코프로세서 스택 레지스터중, ST(0)와 ST(1) 스택 레지스터의 값을 소스로하여 삼각함수 연산을 수행하고 그 결과를 ST(0)에 저장하는 방식을 사용한다. ST(0)에 저장되어 있는 결과 값은 fac(Floating point ACcumulator)라는 매개변수를 통하여 삼각함수를 호출하였던 C함수에 전달하게 된다. ST(0)에 있는 소스 값은 $-2^{63}\sim+2^{63}$범위의 라디안 형태로 주어져야 한다. 이 값의 범위를 넘을 경우 Invalid Operation Exception이 발생하여 FPU ISR 루틴에서 처리하게 된다. 무한대의 값을 입/출력될 경우에도 Invalid Operation Exception이 발생하게 된다. 입/출력값이 $\pm10^{308}$을 넘을 경우 Underflow나 Overflow Exception이 발생하며, FPU ISR루틴을 통해서 처리되어진다. asin과 acos인 경우에는 80387에서 직접 지원하는 명령이 존재하지 않기 때문에, atan함수를 이용한 삼각함수의 공식을 이용하여 asin과 acos이 구현되었다.

구분	80387로 구현된 Library			MSVC Library(80187)		
	Instruction Executed	CPU Cycles	Total Exec Time(us)	Instruction Executed	CPU Cycles	Total Exec Time(us)
sin()	25	108	89.6	98	348	169.8
cos()	25	108	89.6	99	355	170.6
tan()	27	114	88.3	152	550	212.1
asin()	71	272	118.9	130	462	188.8
acos()	82	300	133.1	138	482	199.0
atan2()	104	430	154.0	136	448	194.7

그림 6. Throughput 비교

그림 6에서는 다목적실용위성 2호 RDU의 80387 코프로세서를 위하여 80387 코드로 구현된 삼각함수를 이용한 throughput과 MSVC에서 지원하는 library를 이용한 삼각함수의 throughput을 비교한것이다. sin, cos, tan인 경우에는 거의 2배의 throughput 향상이 있으며, asin, acos, atan2 인 경우 60%이상의 throughput 향상이 있다. 기존 1호기에서 사용하던 PACLIB와 구현된 80387 함수와 비교시에는 8배의 성능향상이 이루어졌다.

3.2. 80387 코프로세서를 이용한 기타 함수 구현

fabs, floor, sqrt, fmod 4개의 함수 또한 80387 코프로세서를 이용하여 구현되어졌으며, ISR루틴의 불필요한 호출을 막기 위하여 sqrt인 경우에는 음수가 입력으로 들어올 경우에는 matherr를 호출하도록 하였다. 아래 그림 7은 위의 함수들을 이용한 자세제어 모듈의 throughput을 보여준다.

	80387함수	80187함수	Paclib함수
자세제어 모듈 ACS_1_3	301.100us	352.500us	1.091ms

그림 7. ACS_1_3을 이용한 throughput 비교

4. RDU를 위한 수치연산 함수의 Library 및 Build

RDU에 사용될 위성탑재 소프트웨어에서 수치연산 함수를 사용하기 위해서는 구현된 수치연산 함수를 library화하여 위성탑재소프트웨어 빌드시 링크 되어져야 한다. MSVC에서 제공하는 math library를 사용하지 않고 모든 수치연산을 지원하기 위해서는 정수연산에 대한 수치연산에 대한 기능도 반드시 제공되어야 한다. RDU build시 사용되는 모든 수치연산 함수에 대하여 분석하게 되면,

1) long integer와 관련 된 모듈 : _aFftol, _aFldiv, _aFlmul, _aFulmul,_aFlrem
2) Coprocessor H/W를 위한 모듈 : _fac, _fltused
3) VRTX에서 제공하는 것 : bfill, bcopy, strcpy, abs
4) 387 math를 위한 모듈 : sin, cos, tan, asin, acos, atan, atan2, floor, fabs, sqrt, fmod
5) sprintf

총 5가지로 구분할 수 있다. 위의 5가지를 모듈중에서 MSVC의 math library에서 의존하고 있는 부분은 long integer와 관련된 부분이다. long integer를 위해서 paclib 에서 제공하고 있는 imath.asm을 이용하여 대체하였으며, sprintf는 1호기와 동일하게 paclib에서 제공하는 라이버리를 이용하여 구성하였다. 그리고, bill, bcopy등과 하드웨어 관련된 부분은 VRTX에서 제공하는 소스를 이용하여 Polymake V3.0를 이용하여 RDU math library를 새롭게 구성하였다. 새롭게 만들어진 RDU library는 clibncp_rdu.lib는 구성되며, RDU 전체 build시 기존에 사용하였던 ssilc7.lib, exe_387.asm, clibncp.lib를 제거하고 이 library만을 링크하여 사용하게 된다. 그림 8는 전체 build시 링크된 모듈과 map파일에서의 위치를 보여준다.

```
i. Modules from clibncp_rdu.lib

C:\KOMPSAT-2\lib\clibncp_rdu.lib(..\c_src\bcopy.c)
C:\KOMPSAT-2\lib\clibncp_rdu.lib(..\c_src\sprintf.c)
C:\KOMPSAT-2\lib\clibncp_rdu.lib(..\asm_src\fltused.a86)
C:\KOMPSAT-2\lib\clibncp_rdu.lib(..\c_src\bfill.c)
C:\KOMPSAT-2\lib\clibncp_rdu.lib(..\asm_src\ftol.a86)
C:\KOMPSAT-2\lib\clibncp_rdu.lib(..\c_src\abs.c)
C:\KOMPSAT-2\lib\clibncp_rdu.lib(..\asm_src\i387.a86)
C:\KOMPSAT-2\lib\clibncp_rdu.lib(..\asm_src\imath.a86)
C:\KOMPSAT-2\lib\clibncp_rdu.lib(..\asm_src\fac.a86)
C:\KOMPSAT-2\lib\clibncp_rdu.lib(..\c_src\strcpy.c)

ii. RDU map
......  ......  .....  .... ..............  ....
3ABD0H  3ACFDH  012EH  WORD UTL_SWA2_TEXT   CODE
3ACFEH  3C62BH  192EH  WORD SCLIB          CODE
3C62CH  3C62CH  0000H  WORD VRTXCODE_TEXT   CODE
3C62CH  3C9D8H  03ADH  BYTE _TEXT           CODE
5D000H  5EEBCH  1EBDH  PARA VRTX_CODE       CODE
70000H  7FFFFH  64K    ABS  _SM_SEG
80000H  8FFFFH  64K    ABS  _LSM_SEG
FFE00H  FFE53H  0054H  WORD EXE_BOOT_TEXT   CODE
FFFF6H  FFFFFH  000AH  WORD EXE_BLD_TEXT    CODE
FFFF0H  FFFF4H  0005H  ABS  (BOOTSTRAP)
```

그림8. RDU Memory Map

5. 결론

본 논문에서는 다목적실용위성 2호의 RDU에 장착된 80387 코프로세서를 위한 수치연산 함수의 구현과 초기화 및 ISR루틴 처리, 그리고 library로 구성하는 방법에 대해서 설명하였다. 현재 RDU의 탑재소프트웨어와 수치연산 함수는 전체 빌드되어 지속적인 검증 시험을 수행하고 있다.

6. 참고문헌

[1] Intel. Intel 80387 Programmer Reference Manual, 1987
[2] Intel. Intel387 DX Math Coprocessor, 1995
[3] Intel. Intel 80387 Programmer Reference Manual, 1986
[4] 최종욱, KARI-SED-TM-2001-004, 다목적실용위성 2호 RDU(Remote Drive Unit)의 Intel 80387 Coprocessor를 위한 수치연산 프로그램 설계, 2001
[5] 최종욱, KARI-SED-TM-2002-002, 다목적실용위성 2호 RDU(Remote Drive Unit)의 Intel 80387 Coprocessor를 이용한 수치연산 함수 구현, 2002