

EA0028: 임베디드 소프트웨어 I 실습

상명대학교 컴퓨터과학과

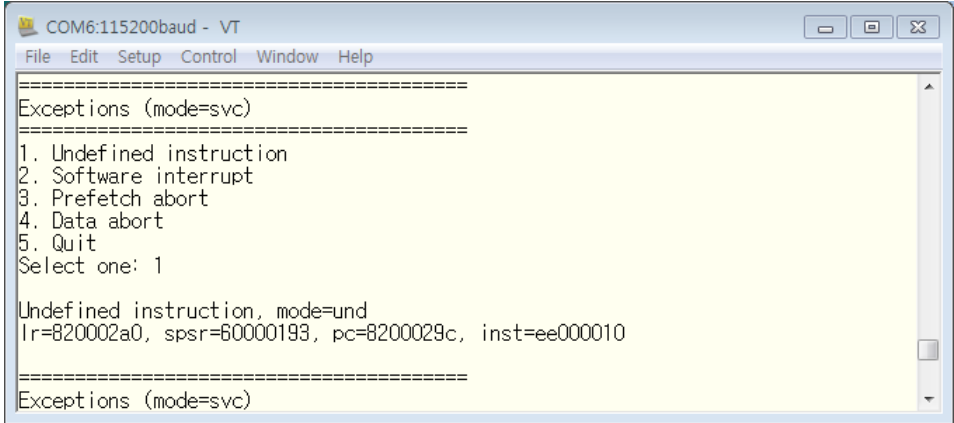
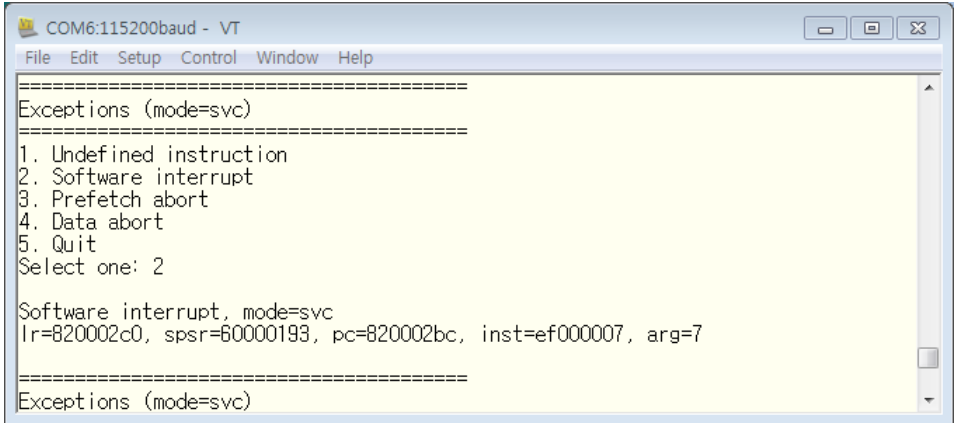
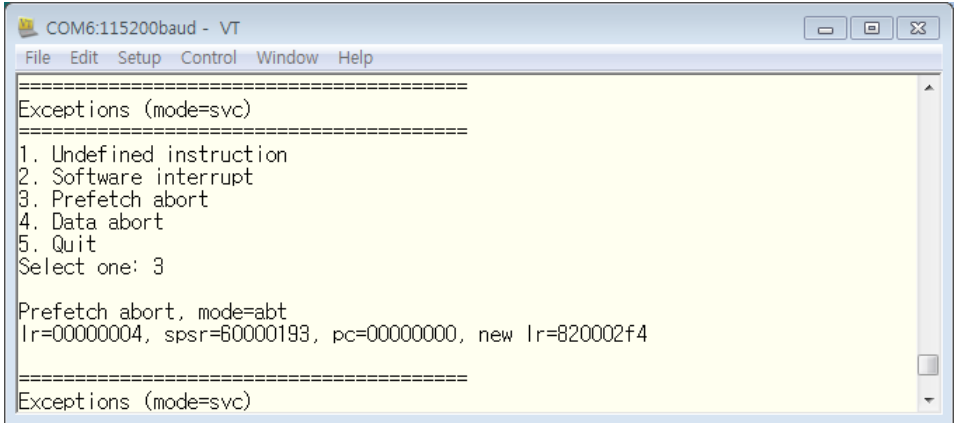
2018년 1학기

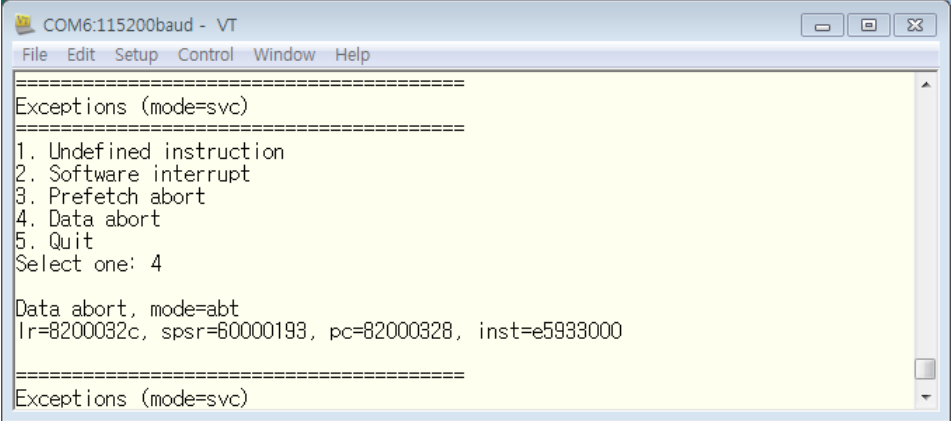
| | | | |
|-------|--|-------|-------------|
| 실습 번호 | 05 | 실습 점수 | /15 |
| 실습 날짜 | 2018년 월 일 | 실습 폴더 | ~/es1/lab05 |
| 학생 이름 | | 학번 | |
| 실습 제목 | Exception Programming | | |
| 참고 자료 | 1. 신동하, 6 Exception Programming, 임베디드 소프트웨어 I 강의 자료, 2018. 2. ARM Limited, ARM® Architecture Reference Manual, ARM DDI 0100I, July 2005. (Chapter A2.6) 3. ARM Limited, ARM® Architecture Reference Manual ARM®v7-A and ARM®v7-R Edition, ARM DDI 0406B, July 2009. (Chapter B1.6) | | |

| | | | |
|-------|--|----|---|
| 실습 번호 | 1 | 점수 | 5 |
| 실습 내용 | <p>ARM exception 처리에 필요한 프로그램 파일 start.S를 assembly 언어로 작성하라. 이 프로그램은 아래와 같이 각 exception mode의 sp를 설정하고, bss section을 초기화한 후, C 함수 main을 부르는 부분과, exception vvector를 정의하고 C 언어로 작성된 각 exception handler 함수(이들 함수는 파일 handlers.c에서 정의함)를 부르는 부분으로 구성된다.</p> <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <ol style="list-style-type: none"> ① 레지스터 lr을 u-boot의 stack(=sp_svc임)에 push한다. ② u-boot의 sp를 변수 _saved_sp에 저장한다. ③ 현재 VBAR 값을 변수 _saved_vector_base_register에 저장하고 새로운 vector 주소를 VBAR에 저장한다. ④ 각 모드(fiq, irq, abt, und, usr, svc)의 stack pointer를 (0x82000000-0x?)로 설정한다. ⑤ bss section을 0으로 초기화한다. ⑥ C 언어 함수 main()을 call한다. ⑦ 변수 _saved_vector_base_register로부터 VBAR을 복구한다. ⑧ 변수 _saved_sp로부터 sp를 복구한다. ⑨ pc를 u-boot의 stack에서 pop하여 얻은 값으로 수정한다 <ol style="list-style-type: none"> ① 새 exception vector를 프로그램한다. ② 각 exception을 처리하는 곳의 시작 주소를 .word로 정의한다. ③ 각 exception 마다 stack에 r0-r12, lr을 저장하고 r0=sp, r1=spsr을 저장하고 C로 작성된 excpetion handler를 call한다. ④ exception에서 돌아오면 stack에서 r0-r12, pc를 복구하여 exception의 처리를 마친다. </div> | | |

| | |
|-------|--|
| 실습 결과 | 1.1. 작성한 프로그램 파일 start.S를 출력하고 출력물 우측 여백에 연필로 직접 프로그램의 설명을 적어서 제출하라. |
|-------|--|

| 실습 번호 | 2 | 점수 | 5 | | | | | | | | | | |
|--------------------------|--|---|---|-----------|-------------------|---------------------------------|---|------------------------------|---|--------------------------|--|----------------------|---|
| 실습 내용 | <p>앞 실습에서 작성한 프로그램 파일 start.S와 연결되어 각 exception을 실제 처리하는 handler 함수들로 구성되는 프로그램 파일 handlers.c를 작성하라. 각 handler 함수가 하는 일은 아래와 같다. 각 함수 프로그래밍 시 2개의 인수를 적절히 사용하여야 하고, 현재 mode를 CPSR을 보고 알아내는 함수 char *get_current_mode(void)를 작성하여야 한다.</p> | | | | | | | | | | | | |
| | <table><tr><th>Exception</th><th>Handler 함수에서 하는 일</th></tr><tr><td>Undefined instruction exception</td><td><p>① 발생한 exception 이름을 출력.</p><p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p><p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p><p>④ exception이 발생한 곳의 PC 및 instruction을 %8x로 출력.</p></td></tr><tr><td>Software interrupt exception</td><td><p>① 발생한 exception 이름을 출력.</p><p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p><p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p><p>④ exception이 발생한 곳의 PC 및 instruction을 %8x로, SWI instruction의 인수를 %d로 출력.</p></td></tr><tr><td>Prefetch abort exception</td><td><p>① 발생한 exception 이름을 출력.</p><p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p><p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p><p>④ exception이 발생한 곳의 PC 및 돌아갈 주소인 new lr을 %8x로 출력.</p></td></tr><tr><td>Data abort exception</td><td><p>① 발생한 exception 이름을 출력.</p><p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p><p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p><p>④ exception이 발생한 곳의 PC 및 instruction을 %8x로 출력.</p></td></tr></table> | | | Exception | Handler 함수에서 하는 일 | Undefined instruction exception | <p>① 발생한 exception 이름을 출력.</p> <p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p> <p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p> <p>④ exception이 발생한 곳의 PC 및 instruction을 %8x로 출력.</p> | Software interrupt exception | <p>① 발생한 exception 이름을 출력.</p> <p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p> <p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p> <p>④ exception이 발생한 곳의 PC 및 instruction을 %8x로, SWI instruction의 인수를 %d로 출력.</p> | Prefetch abort exception | <p>① 발생한 exception 이름을 출력.</p> <p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p> <p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p> <p>④ exception이 발생한 곳의 PC 및 돌아갈 주소인 new lr을 %8x로 출력.</p> | Data abort exception | <p>① 발생한 exception 이름을 출력.</p> <p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p> <p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p> <p>④ exception이 발생한 곳의 PC 및 instruction을 %8x로 출력.</p> |
| | Exception | Handler 함수에서 하는 일 | | | | | | | | | | | |
| | Undefined instruction exception | <p>① 발생한 exception 이름을 출력.</p> <p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p> <p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p> <p>④ exception이 발생한 곳의 PC 및 instruction을 %8x로 출력.</p> | | | | | | | | | | | |
| | Software interrupt exception | <p>① 발생한 exception 이름을 출력.</p> <p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p> <p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p> <p>④ exception이 발생한 곳의 PC 및 instruction을 %8x로, SWI instruction의 인수를 %d로 출력.</p> | | | | | | | | | | | |
| Prefetch abort exception | <p>① 발생한 exception 이름을 출력.</p> <p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p> <p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p> <p>④ exception이 발생한 곳의 PC 및 돌아갈 주소인 new lr을 %8x로 출력.</p> | | | | | | | | | | | | |
| Data abort exception | <p>① 발생한 exception 이름을 출력.</p> <p>② 현재 mode를 CPSR을 보고 알아내어 출력.</p> <p>③ exception 발생 시 저장된 LR 및 SPSR을 %08x로 출력.</p> <p>④ exception이 발생한 곳의 PC 및 instruction을 %8x로 출력.</p> | | | | | | | | | | | | |
| 실습 결과 | <p>2.1. 작성한 프로그램 파일 handler.c를 출력하고 출력물 우측 여백에 연필로 직접 프로그램의 설명을 적어서 제출하라.</p> | | | | | | | | | | | | |

| 실습 번호 | 3 | 점수 | 5 |
|-------|---|----|---|
| 실습 내용 | <p>앞 실습에서 작성한 프로그램 파일 start.S 및 handlers.c와 주어진 프로그램 파일 exceptions.c를 컴파일하여라. 이 프로그램을 수행하고 화면에 나타나는 메뉴 1 - 4를 순서적으로 선택하여 해당 exception을 발생시킨 후 아래와 같은 화면이 나타나는 지를 확인하라.</p> <p>1) Undefined exception 처리 화면</p>  <p>2) Software interrupt exception 처리 화면</p>  <p>3) Prefetch abort exception 처리 화면</p>  | | |

| | |
|--------------|--|
| | <p>4) Data abort exception 처리 화면</p>  |
| <p>실습 결과</p> | <p>3.1. 위 그림과 같이 프로그램 exceptions.bin을 BeagleBone 상에서 수행한 화면을 출력하여 제출하라. (학생의 출력 값은 위 그림과 다를 수도 있음)</p> <p>3.2. 출력한 화면 상에 나타나는 각 숫자가 의미하는 내용을 파일 exceptions.dis에서 찾아서 설명하라.</p> |