

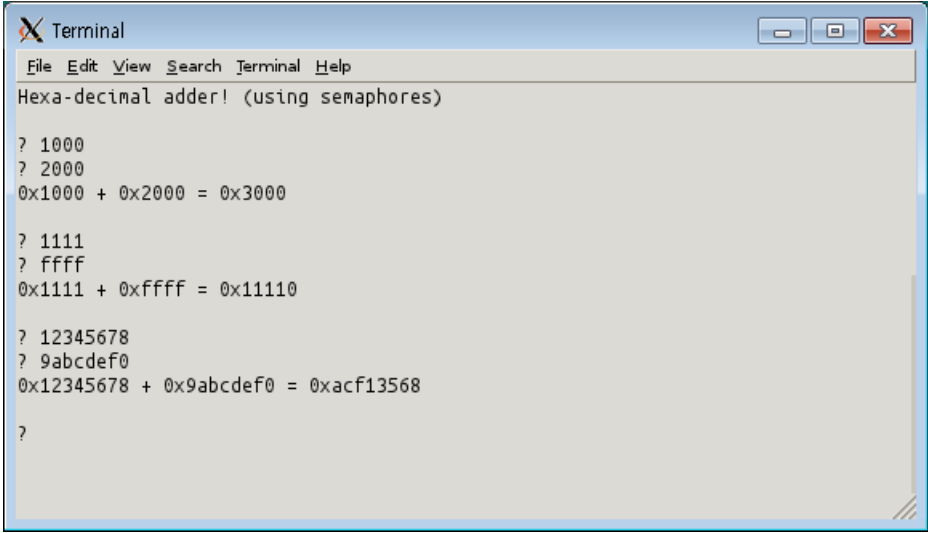
## EA0028: 임베디드 소프트웨어 I 실습

상명대학교 컴퓨터과학과

2019년 1학기

실습 번호	09	실습 점수	/6
실습 날짜	2019년 월 일	실습 폴더	~/es1/lab09
학생 이름		학번	
실습 제목	uC/OS-II Synchronization Programming		
참고 자료	<ol style="list-style-type: none"> <li>1. 신동하, 3 Task Management, 임베디드 소프트웨어 I 강의 자료, 2019.</li> <li>2. 신동하, 4 Time Management, 임베디드 소프트웨어 I 강의 자료, 2019.</li> <li>3. 신동하, 5 Event Control Block, 임베디드 소프트웨어 I 강의 자료, 2019.</li> <li>4. 신동하, 6 Semaphore Management, 임베디드 소프트웨어 I 강의 자료, 2019.</li> <li>5. Jean J. Labrosse, MicroC/OS-II The Real-Time Kernel Second Edition, CMP Books, 2002.</li> <li>6. 성원호, MicroC/OS-II 실시간 커널, 에이콘 출판사, 2005.</li> </ol>		

실습 번호	1	점수	3
실습 내용	<p>BeagleBone에서 다음 조건을 만족하는 실시간 커널 uC/OS-II 프로그램 파일 sync-delay.c를 작성하라.</p> <ul style="list-style-type: none"> <li>• 이 프로그램의 함수 main()에서는 3개의 task인 Task1, Task2 및 Task3을 priority 11, 12 및 13으로 생성한다.</li> <li>• Task1은 UART에 연결된 화면을 clear한 후 "Hexa-decimal adder! (using delay"을 출력한다. 그 후 UART에 연결된 keyboard로부터 2개의 16진수 숫자를 입력 받아 적당한 변수(예: var1 및 var2)에 저장하고 10 tick 시간만큼 delay하는 일을 반복 수행한다.</li> <li>• Task2는 Task1이 저장한 2개의 16진수 숫자(예: var1 및 var2)를 더하여 적당한 변수(예: sum)에 저장하고 10 tick 시간만큼 delay하는 일을 반복 수행한다.</li> <li>• Task3은 Task2가 더한 변수(예: sum)의 값을 16진수 숫자로 UART에 출력하고 10 tick 시간만큼 delay하는 일을 반복 수행한다.</li> <li>• 이 프로그램의 수행 화면은 실습 번호 2의 수행 화면과 유사하다.</li> </ul>		
실습 결과	<p>1.1 이 프로그램에서 Task2 및 Task3이 10 tick마다 계속 수행되지 않고 Task1이 UART에서 입력된 두 16진수를 읽고 난 뒤에만 수행하는 이유를 써라.</p> <p>1.2 작성한 파일 sync-delay.c를 프린터로 출력하고 출력물 우측 빈 여백에 연필로 직접 프로그램의 설명을 적어서 제출하기.</p> <p>1.3 위 그림과 같이 uC/OS-II 프로그램 sync-delay.bin을 BeagleBone에서 수행시킨 후 화면에 나타나는 동작 화면을 캡처하여 첨부하기.</p>		

실습 번호	2	점수	3
실습 내용	<p>BeagleBone에서 다음 조건을 만족하는 실시간 커널 uC/OS-II 프로그램 파일 sync-semaphore.c를 작성하라.</p> <ul style="list-style-type: none"> <li>이 프로그램의 함수 main()에서는 3개의 task인 Task1, Task2 및 Task3을 priority 11, 12 및 13으로 생성한다.</li> <li>Task1은 UART에 연결된 화면을 clear한 후 "Hexa-decimal adder! (using semaphore)"을 출력한다. 그 후 Task3이 post하는 semaphore를 pending하다가 UART에 연결된 keyboard로부터 2개의 hexa-decimal 숫자를 입력 받아 적당한 변수(예: var1 및 var2)에 저장하고 Task2가 pending하는 semaphore를 post하는 일을 반복 수행한다.</li> <li>Task2는 Task1이 post하는 semaphore를 pending하다가 Task1이 저장한 2개의 hexa-decimal 숫자(예: var1 및 var2)를 더하여 적당한 변수(예: sum)에 저장하고 Task3이 pending하는 semaphore를 post하는 일을 반복 수행한다.</li> <li>Task3은 Task2가 post하는 semaphore를 pending하다가 Task2가 더한 변수(예: sum)의 값을 hexa-decimal 숫자로 UART에 출력하고 Task1이 pending하는 semaphore를 post하는 일을 반복 수행한다.</li> <li>이 프로그램의 수행 화면은 다음 그림과 같다.</li> </ul> 		
실습 결과	<p>2.1 이 프로그램이 앞서 작성한 실습 번호 1의 프로그램보다 다른 점이 무엇인지 설명하라. 이 설명 시 실습 1 및 실습 2에서 수행되는 3개의 task의 상태 변화를 자세히 비교하면서 설명하여야 한다.</p> <p>2.2 작성한 파일 sync-semaphore.c를 프린터로 출력하고 출력물 우측 빈 여백에 연필로 직접 프로그램의 설명을 적어서 제출하기.</p> <p>2.3 그림 1.1과 같이 uC/OS-II 프로그램 sync-semaphore.bin을 BeagleBone에서 수행시킨 후 UART에 나타나는 동작 화면을 캡처하여 첨부하기.</p>		