Embedded Operating System Midterm, Chang Gung University, Autumn 2024

Name: Student ID:

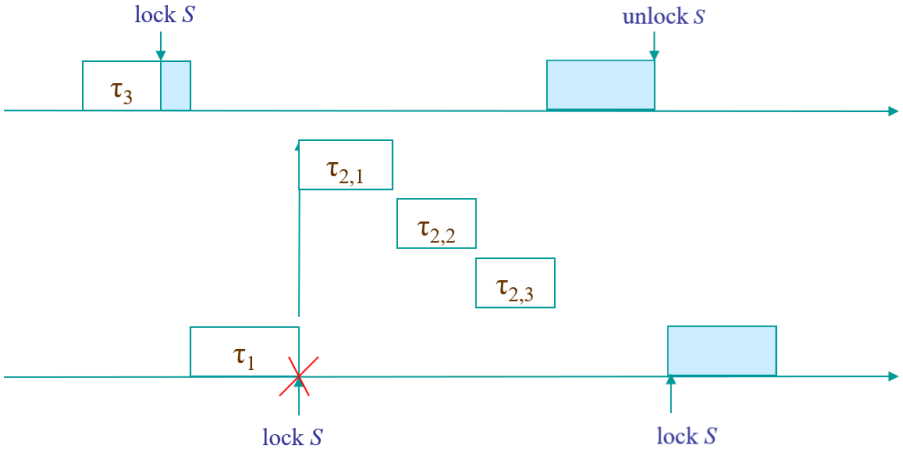
1. (16%) Please define the (a) Non-Recurring Engineering (NRE) Cost and (b) the Unit Cost of a system of products. (c) For a product line with a large number of products, it is more important to reduce the NRE cost or the unit cost? (d) When will the NRE cost be more important than the unit cost?

Answer: (a) Unit Cost: the monetary cost of manufacturing each copy of the system
(b) NRE Cost: the one-time monetary cost of designing the system
(c) The unit-cost reducing is more important for reducing the total cost.
(d) If the number of sales is expected to be small, the NRE cost is more important. It could be some customized products.

2. (8%) To develop software on embedded systems, we usually need the cross-platform development environment consisting of some cross compiler, linker, and source-level debugger. What is the cross compiler? (Where will we run it? The generated binaries are executed on which platform?)

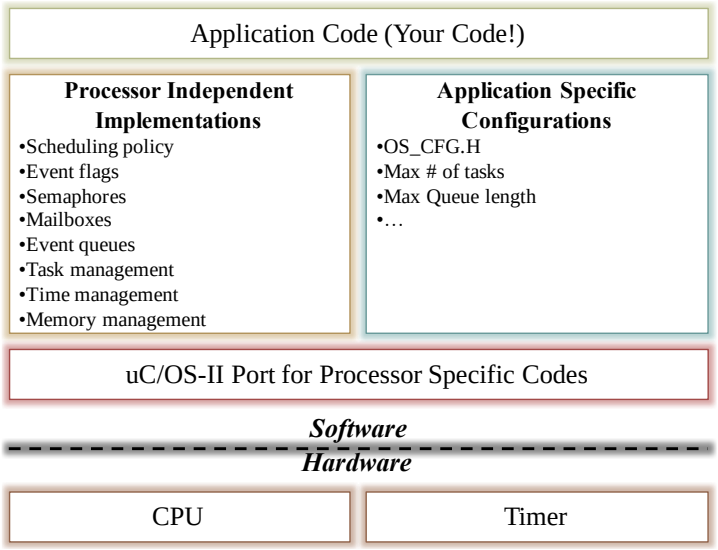
Answer: cross compiler is a compiler which can run on the host system, such as a PC, and can produce the binary which can run on the target embedded system.

3. (16%) (a) Please define Priority Inversion. (b) For the example in the following figure, the three medium-priority tasks arrive when τ_1 is blocked a low-priority task τ_3. Please carefully illustrate the Priority Inversion in the example. (How and where is the Priority Inversion?) (c) Please define Priority Inheritance. (d) How can we solve the Priority Inversion in the example by the Priority Inheritance Protocol?



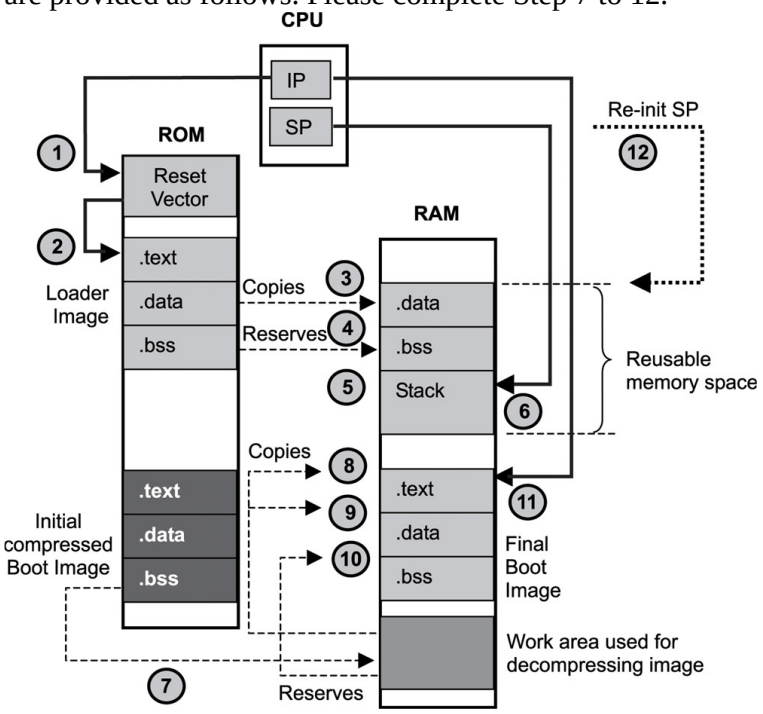
Answer: (a) A high-priority task is (indirectly) preempted by a low-priority task.
(b) When τ_1 is blocked by τ_3, and τ_3 is then preempted by medium-priority tasks, there are priority inversions because when the medium-priority tasks preempt the low-priority task, the high-priority task is also indirectly preempted by the medium-priority tasks.
(c) When a high-priority task is blocked by a low-priority task, the low-priority task temporarily runs with the priority of the a high-priority task.
(d) In the example, when τ_1 is blocked by τ_3, τ_3 runs with the priority of τ_1. Therefore, medium-priority tasks can not preempt the execution of τ_3 when it blocks τ_1. Thus, there is no Priority Inversion.

4. (8%) The following figure shows the structure of µC/OS-II. If now we want to launch a new application on a running system with µC/OS-II, please explain the process for running the new application on µC/OS-II.



Answer: We have to compile the whole package including the OS and application source files, shutdown the system, install the whole image, and reboot the system.

5. (12%) Let's have an example for an image transferred from ROM and running on RAM. Steps 1 to 6 are provided as follows. Please complete Step 7 to 12.



1. The CPU's IP register is hardwired to execute the first instruction in memory, i.e., the reset vector
2. The reset vector jumps to the first instruction of the .text section of boot image
3. The .data section is copied to RAM
4. Reserve space if RAM for the .bss section
5. Reserve stack space in RAM
6. Set SP register to the beginning of the newly created stack

Answer:

7. Copy the Compressed application image from ROM to RAM in a work area
8. Decompress and initialize the application image for instructions
9. Decompress and initialize the application image for global data

10. Decompress and initialize the application image for .bss section
11. The loader transfers control to the image using a processor-specific jump instruction
12. Recycle the memory area occupied by the loader and the work area and reinitialize the SP to point to the memory area occupied by the loader to use it as the stack space

6. (8%) Please define (a) “Soft” real-time systems and (b) “Hard” real-time systems.

Answer: (a) Soft real-time systems: We want to meet the deadline constraint so as to guarantee the quality of applications, but deadline missing is not fatal, e.g., multimedia applications.
(b) Hard real-time systems: If the deadline is missed, critical data are permanently lost or people might get hurt. Thus, it does not allow any deadline missing, e.g., nuclear power plant controllers and anti-lock brake systems.

7. (8%) Context switching is an overhead of task scheduling. Thus, whenever we have a new task scheduling algorithm, we would like to analyze the number of context switching. Stack Discipline is a very useful rule for analyzing the context switching overhead of a task scheduling algorithm. Please provide the definition of Stack Discipline.

Answer: If process A preempts process B, process A must complete before process B can resume.

8. (8%) The following function in Example 1 in the textbook is to create 10 tasks which periodically take a random position and print out their numbers. Please explain the usage the four parameters of function OSTaskCreate(). (The arguments are: Task, (void *)&TaskData[i], &TaskStk[i][TASK_STK_SIZE - 1], i+1 in this case.)

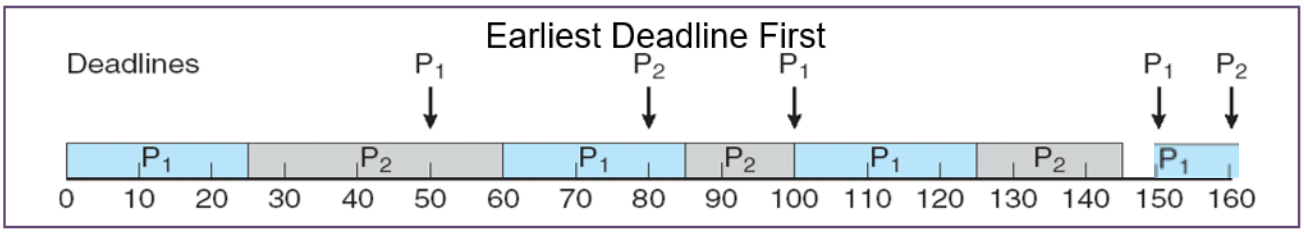
```
static void TaskStartCreateTasks (void)
{
    INT8U i;
    for (i = 0; i < N_TASKS; i++)
    {
        TaskData[i] = '0' + i;
        OSTaskCreate(
            Task,
            (void *)&TaskData[i],
            &TaskStk[i][TASK_STK_SIZE - 1],
            i + 1 );
    }
}
```

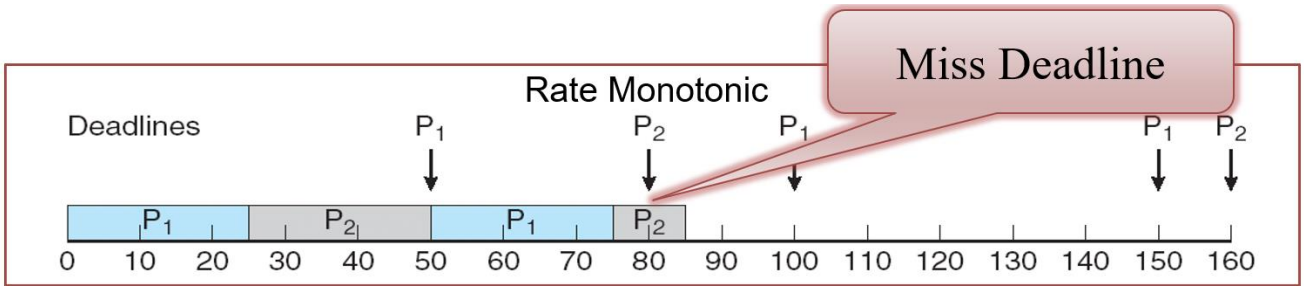
Answer: Entry point of the to-be-created task
The argument for the function of the to-be-created task
The top of stack for the to-be-created task
The priority of the to-be-created task

.

9. (16%) For 2 periodic tasks P_1 and P_2, P_1 has its period 50 and execution time 25, and P_2 has its period 80 and execution time 35. Please draw the scheduling results of (a) the Earliest Deadline First scheduling and (b) the Rate Monotonic Scheduling from time 0 to time 160. If there is any deadline missing, please point it out and stop the scheduling when it has the deadline missing.

Answer:




Miss Deadline
Rate Monotonic
Deadlines
P1
P2
P1
P1
P2
P1
P2
P1
P2
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160