University of Uppsala

Department of Computer Systems (DoCS)

August Examination

23 August 2000

IT3 (Datorsystem II)

Distributed and Real-Time Systems

INSTRUCTIONS TO CANDIDATES

- This is a FIVE (5) hour examination
- Answer all questions on the writing paper provided
- All questions to be answered in English
- Dictionaries are Permitted
- Marks total 85 points
- There are 8 questions
- The exam paper is 4 pages including this cover page

Question 1

Give an example of an inherently sequential program using C style code. Explain why your program is inherently sequential.

[8]

Question 2

Give an example of how asynchronous communication can be useful in a parallel or distributed computer program. Explain one method of implementing a facility to provide asymetric addressed asynchronous communication.

[7]

Question 3

What does the acronym MIMD stand for in the Flynn taxonomy of parallel systems? Explain why some people feel that this class of computer is too general to be useful.

[4]

Question 4

Assume that you can create 1..N parallel copies of a C function using the command $ProcName(1..N, < params >)$; Using this C like parallel programming environment write a parallel function to search through all the values in an array A to find the largest value.
Your algorithm should be able to calculate how to divide up the data evenly among the k copies of the function for any value of k in the range [1..N/2] You can assume that both N and k are powers of 2, that is $N = 2^a$ and $k = 2^b$.
You also need to think about how the copies of the function can know when to stop executing. Explain your termination condition.
Each procedure copy can assume the following interface.

```
void Find_Max(int Id, int A[ ], int N, int k);
```

Where, Id is the identity of the copy of the procedure, A is the Array of values to be searched, N is the number of entries in the array, k is and the number of processes.

[20]

Question 5

Describe two common approaches to deriving realtime schedules. What system characteristics do we need to know? For each system you described comment on its ability to schedule complex systems of tasks and identify the key restrictions on the schedulability measures that are derived from each method.

[8]

Question 6

Assume the following equation is used to compute response time,

$$R_i = C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Given the following system of 4 tasks (without blocking) prove whether or not this system is schedulable using rate monotonic priority ordering.

Task	T	D	C
T_1	50	12	6
T_2	93	30	11
T_3	100	6	4
T_4	110	65	38

[6]

Question 7

Given the following information related to execution costs of Assembler instructions compute a value of C_i for the C code segment given below. You will need to translate the C code into a pseudo-assembler form to perform your analysis.

Instr	Cycles
MOV	8
CMP BGT JMP ADD	4
MUL	16 + 2*(# of set bits in 1st argument)

The code to be analysed is,

```
if ( x > y ) {
    res= x;
} else {
    res= y;
}
⋮
```

[8]

OVER/

Question 8

In a system where blocking can occur, schedules using standard priority assignment and worst case response time calculations often incorrectly predict that a system is schedulable. In this question you will create an example and show how this can happen.

(a). Specify deadlines, periods, computation times and resource sharing activity for a system of three tasks. Use a task vs time diagram to show how blocking on the shared resource combined with standard scheduling causes a priority inversion and thus the high priority task to misses its deadline.

Compute worst case response times for your system of tasks using the standard analysis for deadline monotonic priority assignment without blocking delays. Show that the system of tasks you have proposed is predicted to be schedulable by this analysis.

[8]

(b). Compute the true maximum delays that can be experienced by the tasks in your system by including maximum blocking times and delays due to priority inversion. Show in this way that there are cases where the high priority task will fail to meet its deadline.

[6]

(c). Describe an approach to dealing with priority inheritance that resolves the problem of priority inversion. Using the same example tasks you defined in (a). draw a picture that shows how applying the priority handling approach you have described resolves the priority inversion problem and allows all tasks to meet their deadlines in the worst case situation.

[4]

(d). Finally, use the following equation to compute the worst case response times for the task set you have proposed using the priority handling approach you described in (c). Compute the blocking delay for each task, and then the worst case for each task. Thus, show that your system of tasks is provably schedulable in all cases.

You will need the following formula,

$$R_i = C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

where the blocking factor is defined by

$$B_i = \max_{\{s,t \mid t \in lp(i) \wedge s \in used\_by(t) \wedge ceil(s) \geq pri(i)\}} (cs_{t,s})$$

[6]