Department of Electrical Engineering, IIT Bombay

EE309 Computer Organization, Architecture and MicroProcessors: End-Semester Examination

(Closed book/Closed Notes) Time: 3 hours Maximum Marks: 39

"Thou shalt not covet thy neighbour's answers"

1. **Caustic Stochastix: Considerably Considerate Controller** Consider a computer system with one server, This system is modeled as a variant of the standard M/M/1 system with average service rate μ: there is a controller which controls the entry of jobs into the system, based on the current number of jobs already in the system. If the system has $N-1$ jobs or less, the average arrival rate is λ, as in the original M/M/1 case. For $N+i$ jobs ($0 \leq i < \infty$) in the system, the controller becomes active, and enforces an arrival rate of $\alpha^{(i+1)}\lambda$, $0 < \alpha < 1$. Consider an infinite-length queue, and the other 'standard' Markovian assumptions. Define ρ to be λ/μ.

   (a) Let $p_n(t)$ denote the probability of there being n jobs in the system. Starting from first principles (*ab initio*), derive expressions for the transient response of the system - differential equations for $dp_n(t)/dt$. *(4 marks)*

   (b) Now, derive expressions for the steady-state probability of the system having n jobs, p_n. Your answer should be a function of ρ, p_0 and α. *(4 marks)*

   (c) Is the system stable in the steady state ? Explain, giving the relevant mathematics behind your answer. *(2 marks)*

   (d) Note that with the above controller, the system can reject an arriving job. Derive an expression for the steady-state probability of the system rejecting a job. *(2 marks)*

2. **Shady Pipeline Scheduling**

   (a) Given a *Modified State Diagram* for a static pipeline, devise the most efficient algorithm to enumerate all greedy cycles. *(2 marks)*

   (b) State and explain: The time complexity of your algorithm in terms of the number of states in the Modified State Diagram. *(2 marks)*

   (c) Give an *informal* justification of the correctness of your algorithm. *(1 mark)*

3. **RISCy pipeline: tread carefully !**

```
100 LOAD A, M1
101 ADD A, 1
102 JUMP 104
103 ADD B, A
104 LOAD A, M2
105 ADD A, 1
106 STORE M3, A
107 ADD A, 1
108 JUMP 110
109 ADD B, A
110 STORE M4, A
```

Suppose you were to *manually* generate the best possible optimized code corresponding to the given segment of input code, for a pipelined RISC machine. Explain each point in the code, where you had to make some modification, along with a space-time diagram of the operations in this new piece of code. Clearly mention all line numbers, and write the complete new program, including the statements that

(*continued overleaf*)

would not be executed if execution were to start at line number `100`. Assume that `LOAD` and `STORE` instructions consist of three machine cycles - (i) **I**: Instruction fetch and decode, (ii) **E**: Execution, and (iii) **D**: Memory Operation. All other instructions consist of the first two phases, alone. Here, `M`*i* denote memory locations, and `A` and `B` are registers. The first statement loads `A` with the contents of memory location `M1`; the statement `ADD A, 1` performs `A` ← `A + 1`; etc. *(6 marks)*

4. **All for one, and one for all: Personalized !** Consider the problem of One-to-All *personalized* communication. In a system with p processors (assume p to be a power of 2), processor 0 has an individual message for each of the other $p-1$ processors. Assume Store-and-Forward routing, with a processor sends a packet to another, with one or more messages. A recipient receives a packet, and keeps only the message meant for itself - it re-packetizes one or more of the rest of the messages, and sends them along. Note that while all messages are m words long each, the packet length is obviously, not fixed ! Additionally, note that a processor is permitted to receive *only one* packet from a particular source processor, during the course of the entire operation. Consider solving the problem on two given architectures - a linear ring, and a hypercube.

   (a) Illustrate (diagrammatically) the steps on the two given architectures for $p = 8$, assuming an ideal case of parallelism - processors of equal power, identical interprocessor links, equivalent communications / computations taking equal time, etc. In the diagram, use an arrow to denote the direction of packet transfer, and show the contents of a packet above the arrow, and the time / phase of the particular communication operation, below the arrow. *(2 + 2 marks)*

   (b) Derive an expression for each architecture, for the total time taken, as a function of the setup time (packetization time) t_s, the per-word transfer time along a link t_w, p and the number of words in a message m - for the above ideal case. Assume that the setup time is independent of the packet length (but the transfer time across a link, is not !) *Briefly* explain your answer. *(2 + 2 marks)*

   (c) Now, consider a non-ideal realistic case - write a piece of pseudocode to run on processor number i, for each of the two architectures. Each processor knows its number, the numbers of its adjacent processors, and the total number of processors (p). The command `send(`j`)` is to send a packet to processor number j, and `wait(`j`)` is to receive a packet from processor number j. In order to perform the packetization, a processor is to first call `initialize_packet()`. Then, it attaches the required message (say j: the message from processor 0 meant for processor j) to the packet using `attach_message_to_packet(`j`)`. For example, if a processor has to send a packet consisting of messages 3, 4 and 5 to processor j, it perform the packetization, and the send operation as follows:

```
initialize_packet();
for(k = 3; k <= 5; k++) attach_message_to_packet(k);
send(j);
```

   Do not write any code for a processor to extract the relevant message from a packet that it receives. *(3 + 5 marks)*