COMPUTER ORGANIZATION AND SYSTEM SOFTWARE (COIY060H7)

Coursework reassessment

1. Consider the computation $(m1 + [r1]) * (m2 * r2)$ where $m1, m2$ denote the contents of memory locations, $[r1]$ denotes the content of a memory location whose address is in the register and $r2$ denotes the content of the register.

   (a) Write assembly code typical of RISC machines for this computation. Use at most two operands for each instruction (e.g., `ADD r1 r2` for `r1 <- r1 + r2`) and use at most three registers altogether. (10 marks)

   (b) Identify the dependencies (both true and false) in your code. (10 marks)

   (c) List the various addressing modes in your code. (6 marks)

   (d) Describe an addressing mode suitable for supporting VM (virtual memory). (4 marks)

   (Subtotal: 30 marks)

2. Consider a superscalar processor with two functional units for each of the five pipeline stages: fetch, decode, register read, execute, write back.

   (a) Show the pipeline activity when your code for the previous question is executed using the in-order-issue/in-order-completion policy. State explicitly any additional assumptions you made about processing the instructions. (10 marks)

   (b) Remove the false dependencies from your code by using the register renaming technique. (5 marks)

   (c) Show the pipeline activity when the modified code is executed by using the out-of-order issue/out-of-order completion policy. (10 marks)

   (Subtotal: 25 marks)

3. Consider a computing centre (where there are dozens of hard disks). There are 100 jobs and each of them needs 2 seconds CPU time and each spends 1 second in I/O wait (for reading data from one of the hard disks). Compute the optimal overall runtime for the 100 jobs if they run

   (a) uniprogrammed, (5 marks)

   (b) multiprogrammed. (10 marks)

You can assume that the jobs are independent of each other.

(Subtotal: 15 marks)

4. Consider the following pseudo-code:

```
shared boolean flag[2];
flag[0] = flag[1] = FALSE;
proc(int i) {
  while (TRUE) {
    while (flag[(i+1) mod 2] == TRUE);   // loop till FALSE
    flag[i] = TRUE;
    critical_section;
    flag[i] = FALSE;
  }
}
```

Explain whether this code solves the critical section problem for two processes.

(Subtotal: 15 marks)

5. A system has four processes p_1, p_2, p_3, p_4 and three types of dedicated resources R_1, R_2, R_3. The existence vector is $E = (3, 2, 2)$.

   - Process p_1 holds one unit of R_1 and requests one unit of R_2;
   - Process p_2 holds two units of R_2 and requests two units of R_1 and one unit of R_3;
   - Process p_3 holds one unit of R_1 and requests one unit of R_2;
   - Process p_4 holds two units of R_3 and requests one unit of R_1.

   (a) Compute the availability vector. (2 marks)

   (b) Explain whether the system is deadlocked. (5 marks)

   (c) Determine whether this state of the system is safe. (8 marks)

(Subtotal: 15 marks)