UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2000

MSc in Computing Science
for Internal Students of the Imperial College of Science, Technology and Medicine

# PAPER M2

# ARCHITECTURE AND OPERATING SYSTEMS

Tuesday 9 May 2000, 14:30
Duration: 120 minutes

*Answer THREE questions*

Paper contains 4 questions

1 This question concerns concurrency and synchronisation. Consider the following program fragment, which is part of a multi-user room booking application:

```
procedure BookRoom(rm, time)
  if rooms[rm].isFree[time] then
    rooms[rm].isFree[time] := false
    print("Succeeded")
  else print("Sorry, already booked")
  endif
end BookRoom
```

a Explain what might go wrong if two different users try to make the same room booking at the same time.

b Alyssa P Hacker recommends we use a separate lock for each room, as shown below:

```
procedure BookRoom(rm, time)
  LOCK(rooms[rm].lock)
  if rooms[rm].isFree[time] then
    rooms[rm].isFree[time] := false
    print("Succeeded")
  else print("Sorry, already booked")
  endif
  UNLOCK(rooms[rm].lock)
end BookRoom
```

Can you explain why Alyssa might think this would lead to better performance than using just one lock?

c Suppose some users want to book *pairs* of rooms. We provide a new procedure:

```
procedure BookRoomPair(rm1, rm2, time)
  LOCK(rooms[rm1].lock)
  LOCK(rooms[rm2].lock)
  if rooms[rm1].isFree[time] and rooms[rm2].isFree[time] then
    rooms[rm1].isFree[time] := false
    rooms[rm2].isFree[time] := false
    print("Succeeded")
  else print("Sorry, already booked")
  endif
  UNLOCK(rooms[rm2].lock)
  UNLOCK(rooms[rm1].lock)
end BookRoomPair
```

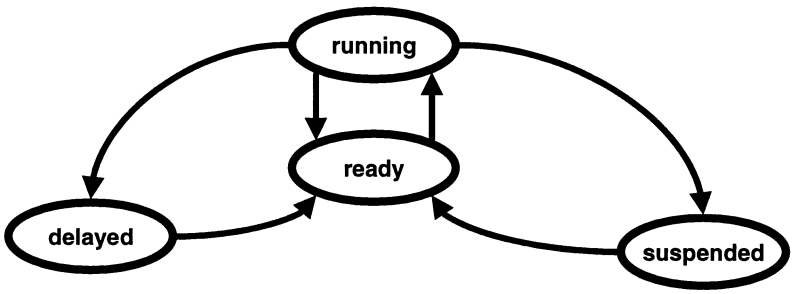
Suppose two users, A and B, attempt to book a pair of rooms at the same time. Using an example, explain what might go wrong, and why.

d Suggest a way to avoid the problem with the procedure in part (c).

*(The four parts carry, respectively, 25%, 25%, 25% and 25% of the marks).*

2 This question concerns processes in a simple operating system kernel supporting multiple processes, semaphores, interrupts and pre-emptive scheduling, as presented in the lecture course.

a Each process in this simple kernel can be in one of four different states, as shown in the diagram below:




(i) What event would cause a process to change from Running to Ready? Why does this happen?

(ii) What event would cause a process to change from Running to Suspended? Why does this happen?

b Give an example of why an interrupt handler might call the `V` operation.

c What would happen if the `P` operation were called in an interrupt handler?

*(The four parts carry, respectively, 25%, 25%, 25% and 25% of the marks).*

3 a An assembler takes a file containing source code and generates an executable (machine code) version. The source code of an 8086 program may include

i CPU instructions

ii Assembler directives

iii BIOS calls

iv Comments

Explain what each is, with an example, and describe how it is processed by the assembler.

b The following hexadecimal digits represent values stored at successive byte locations in the memory of an 8086-based microcomputer

| 2468 | | | | | | | 246F |
|---|---|---|---|---|---|---|---|
| FF | 14 | 88 | 08 | A7 | 13 | 4D | 45 |

Consider the 8086 instructions

```
mov ax,[246c]
add ax, 246A
```

What is the decimal representation of the final contents of ax?

c i The following C function returns the product of two non-negative integers

```
int product(int x,y)
{
    int i = 0;
    int j = y;
    while (j > 0)
    {
        i = i + x;
        j = j - 1;
    }
    return i;
}
```

Write the equivalent subroutine in 8086 assembler. Your solution should use a stack frame, preserve the contents of any registers used, and include **EQU**ate statements and informative comments.

ii The following C function returns a non-negative integer raised to the power of another non-negative integer

```
int power(int m,n)
{
    if (n == 0)
        return 1;
    else return product(m,power(m,n-1));
}
```

Write the equivalent subroutine in 8086 assembler. Your solution should use a stack frame, preserve the contents of any registers used, and include **EQU**ate statements and informative comments.

*The three parts carry, respectively, 30%, 30%, 40% of the marks*

4 a Registers, Main store and Magnetic disks form a memory hierarchy. Some of these levels allow data of different sizes to be stored and accessed. For each of these levels of storage outline the addressing mechanisms used to access the different sizes supported.

b i) Typically, instructions can occupy anything from 2 to 6 bytes of main store depending on the type of instruction and the addressing modes used to specify the operands. The internal instruction register however is of fixed size (say 2 bytes). How does the Basic Machine cycle cope with instructions that are longer than the internal instruction register?

ii) What would be the effect on the micro code of the fetch phase of increasing the width of the external system data bus?

iii) What would be the effect on other parts of the basic machine cycle?

c Consider a microprocessor which supports multiple levels of interrupts. Describe how the control unit responds to an externally generated interrupt which occurs during the handling of a lower level interrupt.

*The three parts carry, respectively, 40%, 30%, 30% of the marks.*