063174

THE UNIVERSITY of York

**Degree Examinations 1997**

# DEPARTMENT OF COMPUTER SCIENCE

# Real-Time Systems and Programming Languages

Time allowed: **Three (3) hours**

Candidates should answer not more than **four** questions

1 (25 marks)

(i) [5 marks] Distinguish between an active and a passive resource in the context of concurrent programming.

(ii) [5 marks] How does Ada support active and passive resources?

(iii) [15 marks] The following code illustrates three interfaces to different semaphore implementations in Ada.

```
package Semaphore_Package is
  protected type Semaphore(Initial : Natural := 1) is
    entry Wait;
    procedure Signal;
  private
    Value : Natural := Initial;
  end Semaphore;
end Semaphore_Package;
```

```
package Semaphore_Package is
  task type Semaphore(Initial : Natural := 1) is
    entry Wait;
    entry Signal;
  end Semaphore;
end Semaphore_Package;
```

```
package Semaphore_Package is
  type Semaphore(Initial : Natural := 1) is limited private;
  procedure Wait (S : in out Semaphore);
  procedure Signal (S : in out Semaphore);
private
  type Semaphore is ...; -- you decide
end Semaphore_Package;
```

Describe how each of the package bodies would be implemented, and discuss the advantages and disadvantages of the approaches.

2 (25 marks)

(i) [10 marks] Describe the POSIX message queue facilities.

(ii) [15 marks] Figure 1 represents a particular process in POSIX.

Figure 1: A particular process in POSIX

Integers are read down message queues `in1`, `in2`, `in3`. The process takes these integers in the order they arrive. Initially, all input integers are output to message queue `out1`. If there is any input down message queue `switch` then output is moved to message queue `out2`. Subsequent inputs down `switch` will change the output queue. Sketch how this process would be implemented using POSIX message queues.

3 (25 marks)

(i) [13 marks] Compare and contrast the Modula-1 and occam2 facilities for device driving.

(ii) [12 marks] Devices on a particular machine are controlled by memory-mapped registers which are 16 bits long. Bit 0 is the least significant bit of a register. The most common types of registers used are: **control and status** registers which contain all the information on a device's status, and allow the device's interrupts to be enabled and disabled. **Data buffer** registers act as buffer registers for temporarily storing data to be transferred into or out of the machine via the device.

A control and status register for a VDU display has the following structure:

```
bits

 11     : Busy             -- set when the device is busy
  7     : Done/ready       -- I/O completed or device ready
  6     : Interrupt enable -- when set enables interrupts
  0     : Device enable    -- set to enable the device
```

The register is located at address hexadecimal AA12.

The structure of the VDU's data buffer register is:

```
bits
 15 - 8  : Unused
  7 - 0   : Data
```

The register is located at address hexadecimal AA14.

Interrupts allow devices to notify the processor when they require service; they are vectored. Interrupts for the VDU display occur through vector location hexadecimal 40.

To send data to the VDU device, the data is placed in the data buffer register then the device and interrupts are enabled. New data can be loaded into the data buffer register once the device indicates that it is ready.

Show how a device driver for the VDU can be implemented in occam2. You should assume that the VDU has a display but no keyboard and that the device is error free. Your solution should buffer data for the user and handle interrupts.

4 (25 marks)

(i) [12 marks] Exceptions can be used as a framework for error recovery. Define the following:

1. a synchronous exception
2. an asynchronous exception
3. an application-detected error
4. an execution environment-detected error

and give examples of

1. a synchronous exception detected by the application
2. an asynchronous exception detected by the application
3. a synchronous exception detected by the execution environment
4. an asynchronous exception detected by the execution environment

(ii) [13 marks] Consider the following Ada code fragment:

```
Error_1, Error_2 : exception;
task Watch;

task Signaller;

protected Atc is
  entry Go;
  procedure Signal;
private
  Flag : Boolean := False;
end Atc;
```

```
protected body Atc is
  entry Go when Flag is
  begin
    raise Error_1;
  end Go;

  procedure Signal is
  begin
    Flag := True;
  end Signal;
end Atc;

task body Watch is
begin
  ...
  select
    Atc.Go;
  then abort
    -- code taking 100 millisecond
    raise Error_2;
  end select;
  ...
exception
  when Error_1 =>
    Put_Line("Error_1 Caught");
  when Error_2 =>
    Put_Line("Error_2 Caught");
  when others =>
    Put_Line("Other Errors Caught");
end Watch;


task body Signaller is
begin
  ...
  Atc.Signal;
  ...
end Signaller;
```

Describe carefully the possible executions of this program fragment. You should assume that context switches between tasks can happen at anytime.

5 (25 marks)

(i) [8 marks] Describe fully the semantics of the Ada requeue facility.

(ii) [7 marks] Explain how the following resource controller protected object allocates resources. Include in your description the order in which resources are serviced.

```
Max : constant Positive := 20;
type Request_Range is range 0..Max;
type Resource is ...; -- details not needed for this question

protected Resource_Controller is
  entry Request(R : out Resource; Amount : Request_Range);
  procedure Free(R : Resource; Amount : Request_Range);
private
  entry Assign(R : out Resource; Amount : Request_Range);
  Free_For_Allocation : Request_Range := Request_Range'Last;
  New_Resources_Released : Boolean := False;
  To_Try : Natural := 0;
end Resource_Controller;


protected body Resource_Controller is
  entry Request(R : out Resource; Amount : Request_Range)
        when Free_For_Allocation > 0 is
  begin
    if Amount <= Free_For_Allocation then
      Free_For_Allocation  := Free_For_Allocation - Amount;
      -- allocate the resource
    else
      requeue Assign;
    end if;
  end Request;
```

```
    entry Assign(R : out Resource; Amount : Request_Range)
       when New_Resources_Released is
    begin
      To_Try := To_Try - 1;
      if To_Try = 0 then
        New_Resources_Released := False;
      end if;
      if Amount <= Free_For_Allocation then
        Free_For_Allocation := Free_For_Allocation - Amount;
        -- allocate
      else
        requeue Assign;
      end if;
    end Assign;


    procedure Free(R : Resource; Amount : Request_Range) is
    begin
      Free_For_Allocation := Free_For_Allocation + Amount;
      -- free resources
      if Assign'Count > 0 then
        To_Try := Assign'Count;
        New_Resources_Released := True;
      end if;
    end Free;
  end Resource_Controller;
```

(iii) [10 marks] How would you modify the above program to service requests according to the priority of the calling task?

6 (25 marks)

(i) [12 marks] The following equation allows the worst-case response time ($R_i$) for task $i$ to be obtained from its computation time ($C_i$) and the computation times and periods ($T$) of higher priority tasks.

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

- Under what assumptions is this equation valid?
- Derive this equation for task $i$ - i.e. show why this equation is valid.
- Explain how this equation can be solved to find $R_i$.
- How is this equation used to prove that all task deadlines are met?

(ii) [5 marks] What modifications are necessary if the deadline of a task is greater than its period?

(iii) [8 marks] Consider the following task set. What is the worst-case response time for Task 5 if rate monotonic priority ordering is used?

| name | $T$ | $C$ |
|---|---|---|
| Task 1 | 6 | 1 |
| Task 2 | 13 | 3 |
| Task 3 | 17 | 3 |
| Task 4 | 40 | 4 |
| Task 5 | 20 | 5 |