

**HERIOT-WATT UNIVERSITY**

---

**SCHOOL OF MATHEMATICAL  
AND COMPUTER SCIENCES**

**COMPUTER  
SCIENCE**

---

**CONCURRENT SYSTEMS**

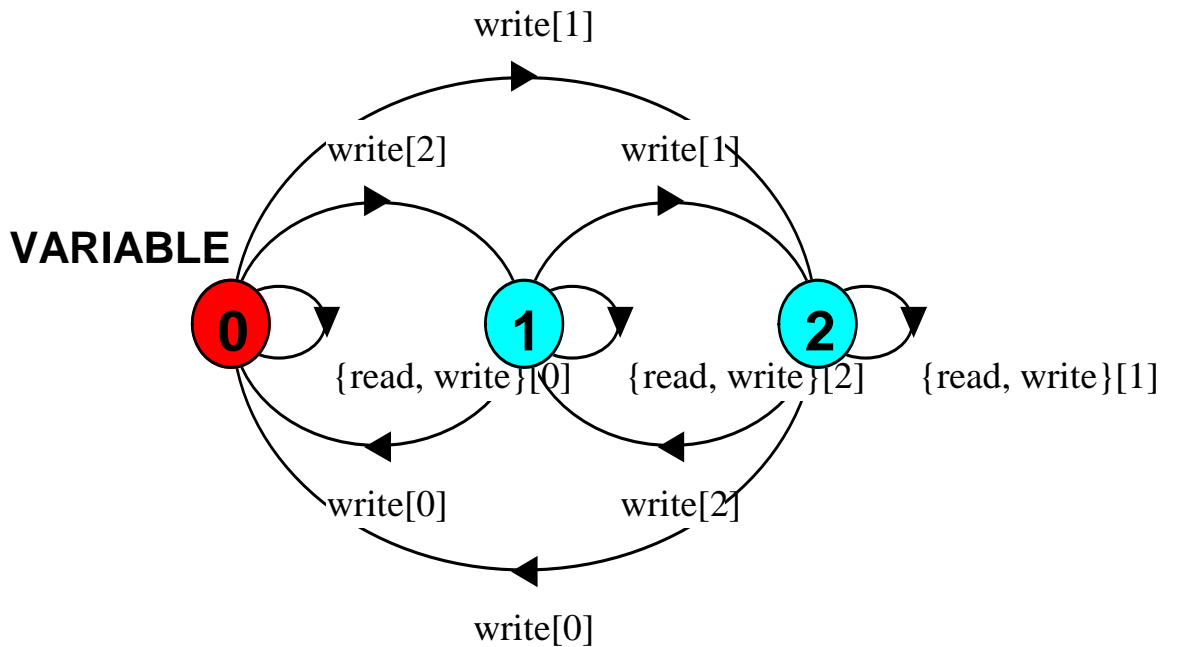
**Wednesday 5 December 2007**

**1.30pm to 3.30pm**

Answer **THREE** questions

**Candidates may only use a University approved  
calculator**

1. (a) In the context of concurrency, explain what is meant by
- trace
  - process
- (5)
- (b) For the process **VARIABLE** described below by the Labelled Transition System (LTS) graph, write down an example of a valid trace.
- (2)



Complete the Finite State Process (FSP) description of the process **VARIABLE**, partially outlined below.

(3)

```
const N = 2
range Int = 0..N
```

```
VARIABLE = VAR[0],
VAR[i:Int] = (read[i]      ->
              |write[v:Int] ->
              ).
```

.../Cont Q1

- (c) Draw LTS diagrams to show whether or not processes **S1** and **S2** below describe the same behaviour.

```

P = (a -> b-> P).
Q = (c -> b-> Q).
||S1 = (P || Q).

S2 = (a->c->b->S2 | c->a->b->S2).

```

(5)

- (d) Write down (in simple English) a description of the following system (given in FSP).

```

const Max = 5
range Int = 0..Max
F(I=0) = P[I],
P[n:Int] = (inc -> P[i+1]
           | when(n>0) dec -> P[i-1]).

```

(5)

2. (a) Explain the difference between a *safety* and a *progress* property in a concurrent system.
- (5)
- (b) Draw a structure diagram for the bounded buffer model **BOUNDEDBUFFER** described in FSP below:

```

BUFFER(N=4) = SPACES[N],
SPACES[i:0..N] = (when(i>0) insert -> SPACES[i-1]
                 | when(i>0) remove -> SPACES[i+1]
                 ).

INSERTION = (insert -> INSERTION).
REMOVAL = (remove -> REMOVAL).

||BOUNDEDBUFFER = (INSERTION | |BUFFER(4) | |REMOVAL).

```

(5)

/Cont...

.../Cont Q2

- (c) Given the bounded buffer described in (b) above, write down (in English) what safety and progress properties are asserted below:

```
property OVERFLOW(N=4) = OVERFLOW[0],
OVERFLOW[i:0..N] = (insert -> OVERFLOW[i+1]
                    | remove -> OVERFLOW[i-1]
                    ).
```

```
|| CHECK_BBUFFER = (OVERFLOW(4) || BOUNDEDBUFFER).
```

```
/* try check with OVERFLOW(3) */
```

```
progress INSERT = {insert}
```

(5)

- (d) How might it be possible to determine whether starvation occurs if item removal from the buffer is lower priority than item insertion.

(5)

3. (a) Why is the metric Million Instructions per Second (MIPS) problematic as a measure for comparing computer performance?

(5)

- (b) Suppose we have two implementations of the same instruction set architecture. Computer A has a clock cycle time of 400 ps and a CPI of 2.0 for a given benchmark program, and computer B has a clock cycle time of 500ps and a CPI of 1.2 for the same program.

- (i) Which computer is faster for this program and by how much?

(3)

- (ii) For this faster computer, a new compiler is released that requires only 0.6 as many instructions as the old compiler but unfortunately it increases the CPI by 1.1. How fast can we now expect the program to run using this new compiler?

(2)

/Cont...

.../Cont Q3

(c) The traditional loop:

```
while (save [i] == k) i = i + j;
```

corresponds to the following code:

```

                                add    $t1, $s3, $s3
                                add    $t1, $t1, $t1
                                add    $t1, $t1, $s6
                                lw     $t0, 0($t1)
                                bne    $t0, $s5, Exit
Loop:                          add    $s3, $s3, $s4
                                add    $t1, $s3, $s3
                                add    $t1, $t1, $t1
                                add    $t1, $t1, $s6
                                lw     $t0, 0($t1)
                                beq    $t0, $s5, Loop
Exit: .....

```

Assume  $i$ ,  $j$  and  $k$  correspond to registers  $\$s3$ ,  $\$s4$  and  $\$s5$  and the base of an array is in  $\$s6$ .

Add appropriate comments and determine the number of instructions required for  $n$  iterations of the loop.

(5)

(d) Ignoring the effects of hazards, consider the following five statements which relate to the efficiency of the MIPS five-stage pipeline given that not all instructions are active in every stage of the pipeline. Which ones are correct?

- (i) Allowing jumps, branches and ALU instructions to take fewer stages than the five required by the load instruction will increase pipeline performance under all circumstances.
- (ii) Trying to allow some instructions to take fewer cycles does not help, since the throughput is determined by the clock cycle; the number of pipe stages per instruction affects latency, not throughput.
- (iii) Allowing jumps, branches and ALU operations to take fewer cycles only helps when no loads or stores are in the pipeline, so the benefits are small.
- (iv) ALU instructions cannot be made to take fewer cycles because of the write-back of the result, but branches and jumps can take fewer cycles, so there is some opportunity for improvement.
- (v) Instead of trying to make instructions take fewer cycles, we should explore making the pipeline longer, so that instructions take more cycles, but the cycles are shorter. This could improve performance.

(5)

4. (a) A processor is partitioned into a pipeline of seven stages. The number of logic levels per stage is 9, 8, 13, 10, 11, 9 and 12 respectively. Clock skew and latch set up times sum to the equivalent of 3 gate delays.
- (i) Determine the clock period for the processor. (2)
- (ii) Given the pipeline from the example above, a designer can divide the third-stage logic path into two stages of 6 and 7 gate delays respectively. Comment upon the wisdom of this design? (3)
- (b) Draw reduced state diagrams for the two pipelines **A** and **B** below. Note that the two pipelines are described by their reservation tables and that pipeline **B** represents pipeline **A** with a delay stage **d** inserted. (5)

Time								Time								
Stage	0	1	2	3	4	5	6	Stage	0	1	2	3	4	5	6	7
1	X			X		X		1	X			X			X	
2		X						2		X						
3			X					3			X					
4					X			4				X				
5							X	5								X
								<b>d</b>					X			

**Pipeline A**
**Pipeline B**

- (c) From the reservation tables, show how the delay insertion in pipeline **B** can improve performance by determining the Minimum Average Latency of the two pipelines. (5)

.../Cont Q4

- (d) Given the control line settings for the **R-format**, **load**, **store** and **branch** instruction types within the 5-stage MIPS processor are as follows:

	RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	
<b>ALUOp0</b>									
<b>R-format</b>	1	0	0	1	0	0	0	1	0
<b>lw</b>	0	1	1	1	1	0	0	0	0
<b>sw</b>	x	1	x	0	0	1	0	0	0
<b>beq</b>	x	0	x	0	0	0	1	0	1

Group the lines by pipeline stage and hence show diagrammatically how the control signals are used in the final three pipeline stages (*ID/EX*, *EX/MEM*, and *MEM/WB*) as the instruction moves down the pipeline.

(5)

**END OF PAPER**