PRIFYSGOL CYMRU; UNIVERSITY OF WALES

DEGREE EXAMINATIONS MAY/JUNE 2003

SWANSEA

Computer Science

CS 213 System Specification
External candidates

Attempt 2 questions out of 3

Time allowed: 2 hours

Students are permitted to use the dictionaries provided by the University

Students are NOT permitted to use calculators

CS_213
SYSTEM SPECIFICATION

(Attempt 2 questions out of 3)

Question 1

Consider the following informal description of a simple microprocessor.

- 64K 16-bit memory;
- Sixteen16-bit user registers, R0 to R15;
- 1-bit condition register C.

Register R15 is used as the program counter. All instructions are 16 bits long, and consequently will fit in a single memory word. The first four bits of an instruction represent the opcode, and the remaining twelve bits represent three registers (four bits for each register). (In the case of the GE instruction, described below, only two of these registers will be used.)

The operations allowed are specified below in a Pascal-like notation, where R, Ra, Rb, D, Da, Db and L represent any of the registers R0 to R15, and Mem is the memory.

ADD D <- Ra, Rb D := Ra+Rb; R15 : = R15 + 1.

NOR D <- Ra, Rb D := not(Ra or Rb); R15 := R15 + 1.

BNC Da,Db,L if C=1 then begin
L := R15+1 ; R15 := Da + Db
end else R15 := R15 + 1.

GE Ra Rb if Ra >= Rb then C := 1 else C := 0 ; R15 := R15 + 1.

LD D <- Ra, Rb D := Mem[Ra + Rb] ; R15 := R15 + 1.

ST R -> Da, Db Mem[Da + Db] := R ; R15 := R15 + 1.

(a) Formally define the *state* of the microprocessor. Your answer should include a clear, formal, description of the *types* of each component of the state. **[5 Marks]**

(b) Formally specify the microprocessor. You should include in your specification all the sub-functions you use. However, it is not necessary to define basic arithmetic and logical operations, or functions for writing to memory and registers. There is an ambiguity in the informal specification, which you should point out. **[15 Marks]**

(c) Extend your formal specification to include a new instruction *unpack*

UNP S, D1, D2

which assumes register S contains two eight-bit characters. The new instruction should extract each of these eight-bit characters, and store them in the least-significant eight bits of registers D1 and D2. The leading, unused, eight bits in each of D1 and D2 should be padded with zeros. **[5 Marks]**

Question 2

(a) Explain, with the aid of the *inner product step processor* (IPS) example from the course notes, the stream transformer representation of simple hardware systems. Your answer should include discussion of *clocks, streams* and *stream transformers*. Note: you are being asked to use the IPS as a supporting example – *not* to simply state the definition of the IPS. **[10 Marks]**

(b) Given a retiming $\square : T \to R$, state the definitions of the immersion $\overline{\square} : R \to T$, the start function $start : \mathrm{Ret}(T, R) \to [T \to T]$, and the length function $l : \mathrm{Ret}(T, R) \to [R \to \mathbf{N}^+]$. **[5 Marks]**

(c) Write down a stream transformer representation of integer counter

$$count : [T \to \mathbf{B}]^3 \to [T \to \mathbf{Z} \times \{u\}]$$

The counter takes three Boolean streams *stsp, ud, rst* as arguments.

The first stream *stsp* starts and stops the counter: if *stsp*(*t*)=*ff* then *count(stsp,ud,rst)(t)=count(stsp,ud,rst)(t-1)*, provided the counter is not being reset (i.e. *rst(t)=ff)*.

The second stream *ud* determines if it counts up or down: if *ud(t)=tt* then *count(stsp,ud,rst)(t)=count(stsp,ud,rst)(t*-1)+1; and if *ud(t)=ff* then *count(stsp,ud,rst)(t)=count(stsp,ud,rst)(t-1)*-1, again provided the counter is not being reset.

The third stream *rst* resets the counter to zero: if *rst(t)=tt,* then *count(stsp,ud,rst)(t)=0* regardless of the values on streams *stsp* and *ud*.

Before the first true element arrives on *rst* stream, the counter's output must be assumed to be *unspecified*.

Write a stream transformer specification of the counter. **[10 Marks]**

Question 3

The following is a (partial) Maude representation of a stack of integers.

```
fmod STACK is
     protecting MachineInt .
     sort StackInt .

     op Empty : -> StackInt .
     op ErrorVal : -> MachineInt .
     op push : StackInt MachineInt -> StackInt .
     op top : StackInt -> MachineInt .
     op pop : StackInt -> StackInt .

     var x : MachineInt .
     var a : StackInt .

     eq top(Empty) = ErrorVal .
     eq pop(Empty) = Empty .
endfm
```

(a) The set of equations in `STACK` is not complete. Write down *two* additional equations, that define the *normal* behaviour of `top`, `pop` and `push`. In addition, in `STACK` no error is reported when the empty stack is popped. Modify the definition of `STACK` so an error is reported if an attempt is made to pop the empty stack. **[10 Marks]**

(b) Write a set of Z schemas to represent a stack of integers. (When applied to the empty stack, your `pop` operation can either return an error or the empty stack: the choice is left to you.) **[10 Marks]**

(c) Compare and contrast the Maude and Z representations. What are the key differences between them? Which do you prefer, and why? **[5 Marks]**

Marking Scheme for CS_213 System Specification 2003 Special Paper

1.(a)

$S = R \times M \times Bit,$

$R = [W_4 \to W_{16}],$

$M = [W_{16} \to W_{16}],$

$Bit = \{0,1\},$

$W_n = Bit^n$

[1 mark] for each.

(b)Here is a basic version with minimal subfunctions (and instruction field extraction functions [1 mark] omitted). It is possible to make this more readable with more liberal use of subfunctions.

$COMP : T \times S \to S,$

$COMP(0,s) = s$

$COMP(t+1,s) = next(COMP(t,s))$

$next : S \to S,$

$next(r,m,c) =$

$$\begin{cases} r[reg1(m(r(15)))/\ r(reg2(op(m(r(15)))) + r(reg3(m(r(15))))][15/\ r(15)+1], m, c & if\ op(m(r(15))) = ADD \\ r[reg1(m(r(15)))/\ r(reg2(m(r(15))))\ nor\ r(reg3(m(r(15))))][15/\ r(15)+1], m, c & if\ op(m(r(15))) = NOR \\ r[reg1(m(r(15)))/\ r(15)+1][15/\ r(reg2(m(r(15)))) + r(reg3(m(r(15))))], m, c & if\ op(m(r(15))) = BNC\ and\ c = 1 \\ r[15/\ r(15)+1], m, c & if\ op(m(r(15))) = BNC\ and\ c = 0 \\ r[15/\ r(15)+1], m, condval(r,m) & if\ op(m(r(15))) = GE \\ r[reg1(m(r(15)))/\ m(r(reg2(m(r(15)))) + r(reg3(m(r(15))))][15/\ r(15)+1], m, c & if\ op(m(r(15))) = LD \\ r[15/\ r(15)+1], m[reg1(m(r(15)))/\ r(reg2(m(r(15)))) + r(reg3(m(r(15))))], c & if\ op(m(r(15))) = ST \end{cases}$$

$condval : R \times M \to C,$

$$condval(r) = \begin{cases} 1 & if\ reg1(m(r(15))) \geq reg2(m(r(15))) \\ 0 & otherwise \end{cases}$$

[2 marks] for the state function COMP, [10 marks] for the next-state function next, and any subfunctions used. [2 marks] for observing the ambiguity in behaviour when the destination register of an operation is R15.

(c) Need to add a new line to the next state function along the lines of

$r[reg1(m(r(15)))/byte1(r(reg3(m(r(15)))))]$ $\quad if \quad op(m(r(15))) = UNP$

$[reg2(m(r(15)))/byte2(r(reg3(m(r(15)))))][15/r(15)+1], m, c$

2.(a) Bookwork exposition on an example discussed in the course. For clocks [2 marks]; streams [3 marks]; stream transformers [2 marks]; for correct defn. of ISP [3 marks]

(b) [1 mark] for immersion; [2 marks each] for start and length

$\overline{\square}(r) = (\square t)[\square(t) = r]$

$start(\square)(t) = \overline{\square}\square(t)$

$l(\square)(r) = \overline{\square}(r+1) \square \overline{\square}(r)$

(c) There are a number of possible solutions. One is

$$count(stsp, ud, rst)(t) = \begin{cases} u & if \ (\square t' \square t)[rst(t') = ff] \\ 0 & if \ rst(t) = tt \\ count(stsp, ud, rst)(t \square 1) & if \ rst(t) = ff \ and \ stsp(t) = tt \\ count(stsp, ud, rst)(t \square 1) + 1 & if \ rst(t) = ff \ and \ stsp(t) = ff \\ & and \ ud(t) = tt \\ count(stsp, ud, rst)(t \square 1) \square 1 & if \ rst(t) = ff \ and \ sts \ p(t) = ff \\ & and \ ud(t) = ff \end{cases}$$

They might choose to build a one cycle delay into the representation, which is acceptable. [2 marks] for each case.

3.(a)Most likely are [3 marks each]

```
eq top(push(a,x))= x .
eq pop(push(a,x))= a .
```

Easiest way to modify representation is to introduce a new constant called (say) ErrorStack [2 marks], and alter the defn. of pop(empty) to pop(empty)=ErrorStack [2 marks].

(b) Based on example discussed in lectures. [4 marks] for preliminary definitions/schemas, [2 marks] each for schemas for push, pop, top.

(c) Variety of possibilities here – most important probably executable nature of Maude/algebraic specs [3 marks] – hence hard to give a complete marking scheme.