Paper Number(s): **E2.7**

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE
UNIVERSITY OF LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2001

EEE PART II: M.Eng., B.Eng. and ACGI

PRINCIPLES OF COMPUTERS AND SOFTWARE ENGINEERING

Friday, 15 June 2:00 pm

There are FIVE questions on this paper.

There are two sections. Answer THREE questions including at least ONE question from each section.

Use a separate answer book for each section.

This is an open book examination.

Time allowed: 2:00 hours

Corrected
copy

Examiners: Shanahan,M.P., Cheung,P.Y.K.

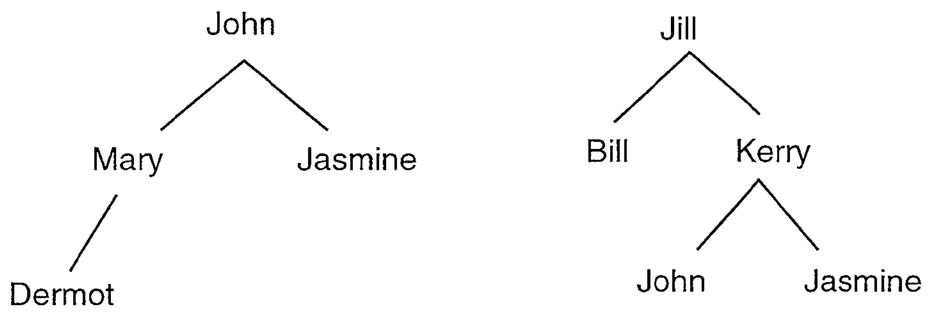
Section A
(Please use a separate answer book for each Section.)

1. Here is the type definition for a binary tree of strings.

```
TTree = ^TNode;
TNode =
record
       Node : string;
       Left : TTree;
       Right : TTree;
end;
```

To answer the following questions, you can assume the existence of access procedures for the type `TTree` called `Empty`, `Left`, `Right`, and `Root` with the obvious meanings.

a) Write a procedure that takes two unordered binary trees of strings T1 and T2 and prints out all the strings that *do* occur in T1 but *do not* occur in T2. For example, suppose T1 and T2 are as follows.



Then the procedure will print out the strings "Mary" and "Dermot".

[10 marks]

b) Suppose T1 contains 10 strings. How much longer will your procedure take to run if T2 contains 7 elements than if it contains 6 elements? Explain your answer.

[4 marks]

c) Modify the procedure so that, in general, it works more efficiently, assuming that the two trees are *ordered.*

Under what circumstances would there be no efficiency gain by using the modified procedure?

[6 marks]

2. Assume the following type definitions

```
type
        TArray1 : array [1..K] of integer;
        TArray2 : array [1..L] of integer;
```

where K and L are declared elsewhere as global constants such that L = 2K. An array A is *ordered* if, for i between 2 and the length of the array, A[i–1] ≤A[i].

a) Write a function that *merges* two ordered arrays into a third. The function should take two ordered arrays A1 and A2 of type TArray1 and return an ordered array of type TArray2 that contains every element occurring in either A1 or A2. Duplicates should not be deleted. For example, if A1 and A2 are as follows,

A1 = [1,3,7,9,13]

A2 = [2,3,8,9,10]

then the function should return the array [1,2,3,3,7,8,9,9,10,13]. Do *not* use a general sort routine to implement your procedure.

[12 marks]

b) What are the maximum and minimum numbers of comparisons *between array elements* the function will perform for any two given arrays. Explain your answer.

[4 marks]

c) How would you modify your function so that the order of the final array is reversed?

[4 marks]

Section B
(Please use a separate answer book for each Section.)

3. The following subroutine, in ARM assembly language, is loaded into memory starting at memory address 0x00008160.

```
1.   ;
2.   ; Subroutine Sqrt - find square root of a 32-bit number
3.   ; Input parameter:  r1 contains a 32-bit unsigned number x
4.   ; Output parameter: r0 contains sqrt(x)
5.   ;
6.   Sqrt   STMED  r13!, {r1-r3, r14}
7.          MOV    r3, #0x8000
8.          MOV    r0, #0
9.   loop   ORR    r0, r0, r3
10.         MUL    r2, r0, r0          ; r2 = r0*r0 (32-bit multiply)
11.         CMP    r1, r2
12.         EORLO  r0, r0, r3
13.         MOVS   r3, r3, ROR #1
14.         BPL    loop
15.         LDMED  r13!, {r1-r3, pc}
```

The **MUL** instruction multiplies two unsigned 16-bit numbers to give a 32-bit product.

This subroutine is called by the following instruction stored at memory address 0x00008134:

```
        BL     Sqrt
```

Just before entering the subroutine `Sqrt`, registers `r1`, `r2`, `r3`, and `r13` contain the following values:

```
r0 = 0xFFFFFFFF      r1 = 0x00000019      r2 = 0xA2470123
r3 = 0x00250032      r13 = 0x00009000
```

a) Draw a diagram showing the addresses and the contents of the stack immediate after the `STMED` instruction is completed.

[4 marks]

b) Describe the operations performed by instructions on lines 12 (**EORLO**) and 13 (**MOVS**).

[2 marks]

c) List the values of registers `r0`, `r1`, `r2`, `r3`, `r15` and the condition code bits (NZCV) for the first 10 instruction cycles after entering the subroutine.

[6 marks]

d) What is the value of register `r0` on exit from the subroutine?

[2 marks]

e) The machine codes for the instructions on lines 12 and 13 are:

```
30200003   EORLO   r0, r0, r3
E1B030E3   MOVS    r3, r3, ROR #1
```

With the aid of diagrams, explain briefly how these two instructions are encoded.

[6 marks]

4. The following two software interrupts are available through the operating system of a microprocessor system:

   `SWI_WriteC` – Write an ASCII character store in register r0 to the console window

   `SWI_ReadC` – Read an ASCII character from the keyboard and return it in register `r0`

   a) Write a subroutine `StringOut` in ARM assembly language for the following specification:

```
; Subroutine StringOut - Output a null-terminated string to console window
;   Input parameters:        none
;   Return parameters:       none
; The String to output must follow the BL instruction immediately.
; For example:
;   .....
;   BL    StringOut
;   =     "Hello World!",0x0a,0x0d,0
;   .....
;   will output on the console "Hello World!" with CR and LF
;
```

[10 marks]

   b) Write a subroutine `HexIn` for the following specification:

```
; Subroutine HexIn - read 8 hex ASCII characters from the keyboard
;                      and convert them to a 32-bit word
; Input parameters:    none
; Return parameters:   r1 contains the 32-bit word entered
```

[10 marks]

5. Consider an 8-bit microprocessor system with a byte addressable main memory space of 2^{16} bytes. 128 bytes of direct mapped cache memory is organised with a cache line size of 16 bytes.

a) How is a 16-bit memory address divided into tag, cache line number and byte number?

[3 marks]

b) The following is a list of memory accesses that occur after power-on reset. What are the cache line numbers used by each of this list of memory accesses? What is the cache miss rate?

Address (hex)	**Read/Write**
000F	R
0080	R
0081	R
008A	W
1047	R
1040	R
0434	R
0435	W
04BF	R
04B0	R
36D6	R
43F4	W
04B1	R
36D0	R

[7 marks]

c) Draw a diagram showing the tag field and the valid bit of the cache memory at the end of this list of memory accesses. Do not show the data fields of the cache memory.

[5 marks]

d) This cache design is to be modified into a two-way set-associative cache with the same amount of cache memory. Draw a simplified diagram of the cache and show how the different fields of the address are interpreted.

[5 marks]

E2.7 SOLUTIONS 2001

Solution to Question 1

a)

[10 marks]

```
procedure TreeDiff(T1,T2 : TTree);
begin
     if T1 <> Empty
     then begin
          if not InTree(Root(T1),T2)
          then writeln(Root(T1));
          TreeDiff(Left(T1),T2);
          TreeDiff(Right(T1),T2);
     end;
end;


function InTree(S : string; T : Tree): boolean;
begin
     if T = Empty
     then InTree := false
     else if S = Root(T)
     then InTree := true
     else InTree :=
          (InTree(S,Left(T)) or InTree(S,Right(T)));
end;
```

b) If T1 contains 10 strings and T2 contains 6 strings, then the procedure will take 10*6*C = 60C time units for some constant C, since the procedure has to traverse T2 once for each member of T1. If T2 contains 7 strings, it will take 70C time units, ie 1.167 times as long (one and one sixth).

[4 marks]

c) The procedure InTree should be replaced by the following code.

```
function InTree(S : string; T : Tree): boolean;
     begin
          if T = Empty
          then InTree := false
          else if S = Root(T)
          then InTree := true
          else if S > Root(T)
          then InTree := InTree(S,Right(T))
          else InTree := InTree(S,Left(T));

     end;
```

There will be no efficiency gain using this procedure if every node in T2 has only one child, ie: T2 is tall and thin.

[6 marks]

Solution to Question 2

a)

```
function Merge(A1,A2 : TArray1): TArray2;
var A3 : TArray2;
begin
      I1 := 1; I2 := 1;
      for I3 := 1 to L do
      begin
            if (I1 ≤ K) and (I2 ≤ K)
            then begin
                  if A1[I1] ≤ A2[I2]
                  then begin
                        A3[I3] := A1[I1];
                        I1 := I1+1;
                  end
                  else begin
                        A3[I3] := A2[I2];
                        I2 := I2+1;
                  end;
            end
            else if I1 ≤ K
            then begin
                  A3[I3] := A2[I2];
                  I2 := I2 + 1;
            end
            else begin
                  A3[I3] := A1[I1];
                  I1 := I1 + 1;
            end;
      end;
      Merge := A3;
end;
```

[12 marks]

b) Maximum number of comparisons is 2K–1. This occurs if the end of one array is reached just before the end of the other array. Minimum number of comparisons is K. This occurs if the whole of one array is before the other.

[4 marks]

c) For loop simply counts down from L to 1 instead of up from 1 to L.

[4 marks]

Solution to Question 3

This question tests students' ability to walk-through a simple assembly language program with good understanding of: addressing modes, stack operations, assembly language syntax, arithmetic operations, function of the barrel-shifter, subroutine calls etc..

a)

[4 marks]

The stack looks like this:

	Addr	**Value**
	0x00009000	0x00008138
	0x00008ffc	0x00250032
	0x00008ff8	0xA2470123
	0x00008ff4	0x00000019
r13 -------------→	0x00008ff0	(previous value – unchanged)

b)

[2 marks]

```
EORLO   r0, r0, r3          ; if carry cleared, then r0 := r0 EXOR r3
MOVS    r3, r3, ROR #1      ; r3 := r3/2 (unsigned), set the status flags
```

c) This shows the contents of registers AFTER each instruction:

[6 marks]

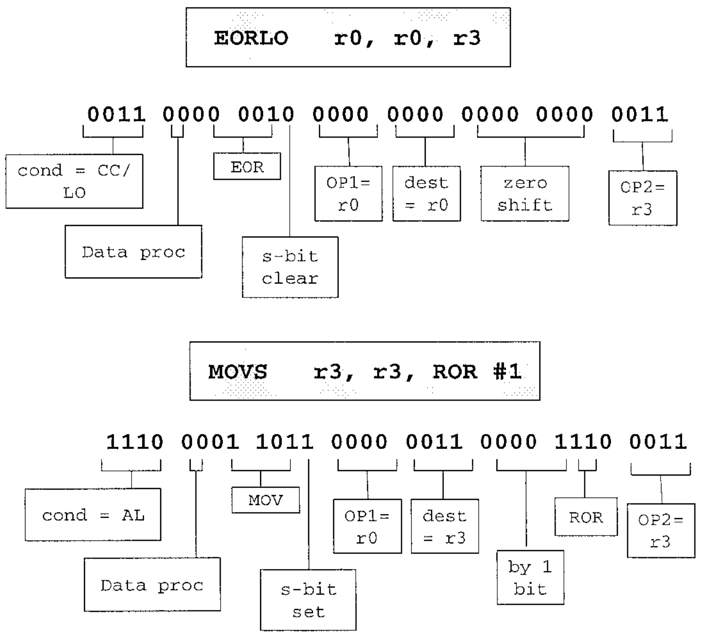
Instr. Cycle	r0	r1	r2	r3	r15	status
1. STMED	FFFFFFFF	00000019	A2470123	00250032	00008164	nzcv
2. MOV	---	---	---	00008000	00008168	nzcv
3. MOV	0	---	---	---	0000816C	nzcv
4. ORR	00008000	---	---	---	00008170	nzcv
5. MUL	---	---	40000000	---	00008174	nzcv
6. CMP	---	---	---	---	00008178	Nzcv
7. EORLO	00000000	---	---	---	0000817C	Nzcv
8. MOVS	---	---	---	00004000	00008180	nzcv
9. BPL	---	---	---	---	0000816C	nzcv
10. ORR	00004000	---	---	---	00008170	nzcv

d) 0x00000005 (sqrt(25))

[2 marks]

3 (continued)

e)



[6 marks]

Solution to Question 4

(a)

[10 marks]

```
          EXPORT    StringOut
; Subroutine StringOut - Output a null-terminated string to console window
; Input parameters:  none
; Return parameters:   none
; The String must follow immediate the BL instruction. For example:
;   .....
;   BL     StringOut
;   =   "Hello World!",0x0a,0x0d,0
;   .....
;   will print "Hello World!" with CR and LF
;
SWI_WriteC EQU        0                  ; Write_C System Call
          AREA       StringOut_src, CODE, READONLY ; name this block of code
StringOut STMED      r13!, {r0}          ; save working registers on stack
Loop      LDRB       r0, [r14],#1        ; get one byte
          CMP        r0, #0
          BEQ        null                ; if null character, terminate
          SWI        SWI_WriteC          ;  else output
          B          Loop                ; repeat until null character found
null      LDMED      r13!, {r0}          ; retrieve registers from stack
          MOV        pc, r14             ;   … and return to calling program
          END
```

(b)

[10 marks]

```
          EXPORT   HexIn
; Subroutine HexIn - read 8 hex ASCII characters and convert to 32-bit word
; Input parameters: none
; Return parameters:  r1 contains the 32-bit word entered
SWI_ReadC EQU      4                     ; Read_C System Call
          AREA     HexIn_src, CODE, READONLY   ; name this block of code
HexIn     STMED    r13!, {r0, r2, r14}  ; save working registers on stack
          MOV      r2, #8          ; r2 has nibble (4-bit digit) count = 8
          MOV      r1, #0                ; r1 has the number, intialize to 0
Loop      SWI      SWI_ReadC             ; Read a character
          CMP      r0, #'0'              ; If char is between '0' and '9'
          BLO      not_num
          CMP      r0, #'9'
          BHI      not_num
          SUB      r0, r0, #'0'          ;   find its value
          B        next_digit
not_num   CMP      r0, #'A'              ; else if char is between 'A' to 'F'
          BLO      not_alpha
          CMP      r0, #'F'
          BHI      not_alpha
          SUB      r0, r0, #'A'-10
          B        next_digit
not_alpha MOV      r0, #0                ; else not legal, set value to 0
next_digit ADD     r1, r0, r1, LSL #4    ; merge the nibble
          SUBS     r2, r2, #1            ; decrement nibble count
          BNE      Loop                  ; if more, do next nibble
          LDMED    r13!, {r0, r2, pc}    ; retrieve registers from stack
                                         ;   … and return to calling program
          END
```

Solution to Question 5

(a)

[3 marks]

15		0
9-bit	3-bit	4-bit
tag	cache line no.	byte no.

(b)

Address (hex)	**Read/Write**	**Cache line no**	**Hit/miss**
000F	R	0	M
0080	R	0	M
0081	R	0	H
008A	W	0	H
1047	R	4	M
1040	R	4	H
0434	R	3	M
0435	W	3	H
04BF	R	3	M
04B0	R	3	H
36D6	R	5	M
43F4	W	7	M
04B1	R	3	H
36D0	R	5	H

Miss rate is 46.7%

[7 marks]

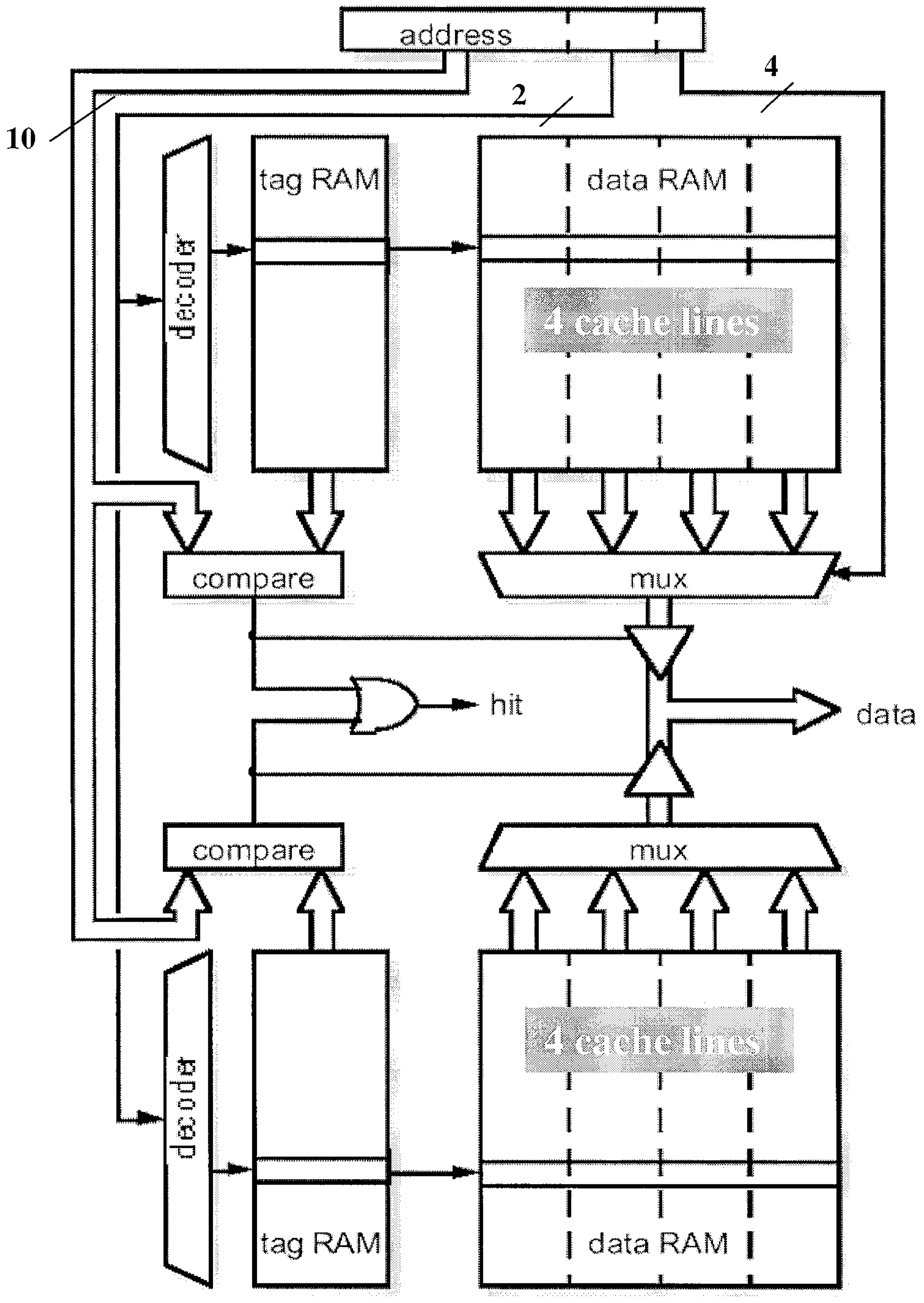
(c)

Cache line	**Valid bit**	**Tag**
0	1	0000 0000 1
1	0	xxxx xxxx x
2	0	xxxx xxxx x
3	1	0000 0100 1
4	1	0001 0000 0
5	1	0011 0110 1
6	0	xxxx xxxx x
7	1	0100 0011 1

[5 marks]

5 (cont.)

(d)



[5 marks]