UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2000

BEng Honours Degree in Computing Part I
MEng Honours Degrees in Computing Part I
BEng Honours Degree in Mathematics and Computer Science Part I
MEng Honours Degree in Mathematics and Computer Science Part I
for Internal Students of the Imperial College of Science, Technology and Medicine

*This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute*
*This paper is also taken for the relevant examinations for the Associateship of the Royal College of Science*

# PAPER C110=MC110

# ARCHITECTURE I

Wednesday 3 May 2000, 14:00
Duration: 90 minutes
(Reading time 5 minutes)

*Answer THREE questions*

Paper contains 4 questions

1a Explain the term *unaligned access* when applied to main memory. Briefly describe how an unaligned word is

i) read from main memory.
ii) written to main memory.

b A Turing-like program has the following data and code:

```
type Data =
    record
        alpha : int_2C       % 16-bit two's complement
        beta  : array 1..3 of char  % ASCII chars
        gamma : int_SM       % 32-bit sign & magnitude
        delta : int_X291     % 24-bit excess 291
    end

var Value : Data

Value.alpha := -292
Value.beta  := "FED"         % "D" is 44 in hexadecimal
Value.gamma := Value.alpha - 3
Value.delta := 0
```

Convert each of the fields of `Value` (after the assignments) to their binary values and rewrite the values in hexadecimal (values should be of the correct size in bytes).

For parts c and d, memory should be shown byte-by-byte, with each byte clearly labelled with its byte address. Assume that memory words are 32-bits wide and memory is byte addressable. Unaligned accesses are supported; so there must be no gaps between fields.

c Show the memory layout of the variable `Value` at the end of the program if run on a:

i) little-endian architecture (LITTLE) with `Value` located at address 1008H.
ii) big-endian architecture (BIG) with `Value` located at address 1008H.

d Show the memory layout of the variable `Value` on BIG if it is copied from `Value` on LITTLE (after the assignments) using:

i) a byte-by-byte transfer
ii) a word-by-word transfer, where words are 32-bits wide.

e Comment on why the results in part d (i) and d (ii) are incorrect (differ from c (ii)), and outline a solution to correctly transfer the variable between LITTLE and BIG.

*The five parts carry, respectively, 20%, 20%, 20%, 20%, and 20% of the marks.*

2a Identify 4 differences between a procedure and an interrupt handler.

b Explain why a DMA controller has priority over the CPU when both request a memory transfer.

c For this part of the question you are asked to develop some procedures that produce a sequence of one second beeps given the following devices:

| | 8 bits | Address |
|---|---|---|
| *Timer Control Port* | | FF0080H |
| *Sound Control Port* | | FF0090H |

Writing a 1 to the Timer Control Port causes a single interrupt to occur in 10 milli-seconds time. Writing a 1 to the Sound Control Port causes a beep to be generated on the speaker.

For this device write the following routines in Pentium assembly language:

```
procedure StartBeeps     % Starts beeps every second

procedure StopBeeps      % Stops beeps

procedure TimerHandler   % Interrupt handler

function  NextBeep:int   % Milliseconds to next beep
```

Once started with `StartBeeps`, beeps should continue to be generated every second until a call is made to `StopBeeps`.

You can assume that the interrupt handler (`TimerHandler`) has been correctly installed in the Interrupt Descriptor Table.

State any additional assumptions that you make.

*The three parts carry, respectively, 20%, 20%, and 60% of the marks.*

3 For this question you are asked to translate the following high-level language procedure and call into Pentium assembly language:

```
procedure BasePrint (number:int, base:int)
var remainder : int
begin
  if number > base then
    BasePrint (number div base, base)
  end if
  remainder := number mod base
  PrintDigit (remainder)
end BasePrint

BasePrint (12, 8)
```

You should assume the following:

- a flat memory model for segments
- 16-bit integers
- that procedures preserve all registers, except register EAX
- instruction CDQ converts the doubleword in EAX into a quadword in EDX:EAX
- that the procedure `PrintDigit` is pre-defined as:

```
procedure PrintDigit (var digit:int)
```

For your answer provide the following:

a a translation of the procedure call:

```
BasePrint (12, 8)
```

b the stack frame just before the **if** statement is executed for the first time, clearly labelling each value on the stack with its offset from the frame pointer.

c a commented Pentium assembly language translation of the procedure `BasePrint`.

d the stack frame just after the first CALL instruction for `PrintDigit` is executed given that the outermost call to `BasePrint` is `BasePrint (12, 8)`. You should show all values on the stack from the outermost call to the CALL instruction for `PrintDigit`.

Note: the parameter to `PrintDigit` is a **var** parameter and the first call of `PrintDigit` occurs in a recursive call of `BasePrint`.

State any additional assumptions that you make.

*The four parts carry, respectively, 10%, 10%, 60%, and 20% of the marks.*

4a For IEEE single-precision, define the representations for infinity values and NaN values. Give an example when each would occur.

b Suppose that the IEEE defined a new 16-bit floating point format called Tiny Precision that follows the same general rules as IEEE Single Precision format except that the Exponent is 6 bits and the Significand is 9-bits

| | 1 bit | 6 bits | 9 bits |
|---|---|---|---|
| *Tiny Precision Format* | Sign S | Exponent E | Significand F |

For this format interpret the 16-bit hexadecimal value CBF1 as a Tiny Precision number and then convert the Tiny Precision number to decimal.

c Convert the decimal number `-130.25` to Tiny Precision and give the result in hexadecimal.

d Perform a floating-point subtraction of the Tiny Precision numbers in parts b and c (subtract the number in part c from the number in part b). Write the result in hexadecimal. Assume that the CPU has additional precision bits to perform the subtraction accurately.

e When the following high-level language statement is executed:

```
print 2.5, 2.6
```

the system prints:

```
2.5       2.59999999999999
```

Explain why the first number is correctly printed, while the second isn't.

*The five parts carry, respectively, 20%, 20%, 20%, 20% and 20% of the marks.*