IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2018

EIE PART II: MEng, BEng and ACGI

Corrected copy

COMPUTER ARCHITECTURE

Monday, 14 May 2:00 pm

Time allowed: 1:30 hours

There are TWO questions on this paper.

Answer TWO questions.

Answer ALL questions.

Use separate answer books for Sections A and B.

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible	First Marker(s) :	C. Thomas,
	Second Marker(s) :	S. Ben Ismail,

1 a i) What is the main goal of pipelining a CPU?

ii) Give one example of a MIPS ISA feature which shows that MIPS CPU implementations were expected to be pipelined.

iii) Construct a plausible *quantitative* example (i.e. containing numbers) showing how pipelining can make a CPU worse. Invent numbers for parameters such as clock-rate, IPC, and so on as needed.

b Consider the following MIPS assembly sequence:

```
f:
  addi $sp, $sp, -4
  div $a0, $a1
  mfhi $v0
  beq $a1, $0, g
  addi $sp, $sp, +4
  jr $ra
  nop
```

i) What is the high-level purpose of the `beq` instruction in this assembly sequence – why would a human or a compiler have inserted it?

ii) Give an example of an exception that could occur while executing the sequence, and the necessary conditions for the exception to occur.

iii) What is the range of addresses that the linker could use for symbol `g`? Show your working.

iv) Optimise this sequence while maintaining functionality, and give the number of cycles needed to execute it in a single-cycle MIPS CPU. State any assumptions you make.

The two parts carry, respectively, 40%, and 60% of the marks.

2 Consider the following C function:

```
int f(unsigned N, unsigned M, int *x)
{
  int y=0;
  // Begin: code of interest
  for(unsigned i=0; i<N; i++){     // outer loop
    for(unsigned j=0; j<M; j++){   // inner loop
      y+=x[j*N+i];
    }
  }
  // End: code of interest
  return y;
}
```

a While manually converting the function to MIPS-1 assembly, an embedded systems designer first optimised the inner loop of the C code to:

```
for(int j=M-1; j>=0; j--){
```

Why did they view this as an optimisation?

b Describe the data locality of the function when using a multi-word set-associative cache in the following scenarios:

i) $N = 1, M >> 1$ ($A >> B$ means "*A much greater than B*")

ii) $N >> 1, M = 2$

iii) $N >> 1, M >> 1$

c i) Assume a static predictor that predicts backwards branches as taken, and forwards branches as not taken. What proportion of branches will be correctly predicted in the function body (denoted "code of interest" in the source)? Include any assumptions and your reasoning.

ii) Sketch a top-level interface of a *dynamic* branch prediction circuit, showing the key input and output signals. Briefly describe the signals.

iii) Design and sketch the implementation of a 4-entry 1-bit dynamic branch predictor in terms of logic and state. You do not need to include logic to update the predictor state, only to make the prediction.

The three parts carry, respectively, 10%, 30%, and 60% of the marks.