UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2004

BEng Honours Degree in Computing Part II
MEng Honours Degrees in Computing Part II
MSc in Computing Science
BEng Honours Degree in Information Systems Engineering Part II
MEng Honours Degree in Information Systems Engineering Part II
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

PAPER C210=E2.13

ARCHITECTURE II

Tuesday 11 May 2004, 14:30
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators required

Section A (Use a separate answer book for this Section)

1 a Provide the diagram of a circuit that generates a single token for initialising implementations produced by a hardware compiler based on the token-passing method.

b Explain how an assignment statement and a concurrent assignment statement can be implemented in hardware using the token-passing method. Both should take a single cycle to complete. Your circuit diagrams should clearly indicate the *start* input for the token to enter, and the *finish* output for the exit of the token.

c Describe, with the use of a circuit diagram, how an assignment statement that takes N cycles where $N > 1$ can be implemented in hardware using the token-passing method. Explain the benefit of this approach.

d Describe, with the use of a circuit diagram, how a WHILE-loop can be implemented in hardware using the token-passing method. Explain how a FOR-loop can be implemented in hardware using a WHILE-loop.

e Provide the diagram of a circuit for the following program, based on the token-passing compilation method:

```
int_8 X
int_8 Y
SEQ
  X := 1
  FOR Y=1 TO 3
    X := X+Y
```

Your solution should minimise the number of cycles.

The five parts carry equal marks.

2a Show how a halfadder can be built using a two-input and-gate and a two-input xor-gate, and show how a fulladder can be implemented using halfadders.

b Design a circuit that takes as input an N-bit unsigned number X and a 1-bit number Y and produces $X + Y$. It must not contain fulladders. Draw a diagram for this circuit for $N = 4$.

c Design a circuit that counts the number of one's in an N-bit number. For instance given 10110, it produces 011. Your design must not contain fulladders. Draw a diagram for this circuit for $N = 4$.

d Design a circuit, containing N fulladders, that takes as input an N-bit unsigned number and multiplies it by 5. Draw a diagram for this circuit for $N = 4$.

The four parts carry, respectively, 20%, 20%, 30%, and 30% of the marks.

Section B (Use a separate answer book for this Section)

3a Your objective is to execute the following program as fast as possible, ignoring virtual memory effects.

```
int B[X],i,Sum=0;  /* 32 bit ints */
B[0]=0; B[1]=X;
for(i=2;i<X;i++){ /* loop1 */
  B[i]=B[i-1]+B[i-2];
}

for(i=0;i<X;i++){ /* loop2 */
  Sum = Sum + B[i];
}
```

i) Assume X=1000, and you can pick your first level cache, specify the minimum size of a direct-mapped cache in order to minimize cache misses and execution time.

ii) Rewrite the program above to minimize the size of the optimal cache. How large a cache do we need now?

b Assume direct mapped caches with 128 byte cache blocks (lines), 8Kbytes first level cache with 1 clock cycle access time, a second level cache of size 1MByte and 10 clock cycles access time and 100 clock cycles to access main memory. Assume you need to execute loop1 in part (a) 1,000 times for each X in [0, 300000]. Draw a graph with X on the x-axis, and expected memory access time on the y-axis.

c Suggest ways to redesign the caches in part (b), to improve expected execution time for different values of X.

The three parts carry, respectively, 40%, 40%, 20% of the marks.

4a Assume a 32 bit virtual memory space, direct-mapped TLB with 8 byte blocks, total size of 128 entries (addresses), and a page size of 1Mbyte. Draw a figure representing the TLB and show which bits of the virtual byte address go to index and tag ports and how the physical address in the case of a machine with 128MByte main memory is generated. Show the bit-widths for all fields and addresses. What is the total size of the TLB implementation in bits? How does this TLB exploit spatial locality?

b Show the final state of the TLB described in part (a) after the following read and write accesses. Show only the relevant lines of the TLB. Accesses are given in the following format: rd(hex word address) and wr(hex word address, value) rd(0x12345678), wr(0x9abcdef0, 0x11223344), rd(0x12345680), rd(0x9abcd888), wr(0x11223355, 0x12345678), rd(0x11223300).

c Usually caches are indexed by the physical address. How about building a memory system where the cache is indexed by the virtual address? List an advantage and a disadvantage of this approach.

The three parts carry, respectively, 50%, 30%, 20% of the marks.