UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2007

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

COMPUTER ARCHITECTURE

Wednesday 2 May 2007, 14:30
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators required

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

1a Memory System

Explain the purpose of cache lines. What happens when you increase the cache line size? What happens when you decrease the cache line size? Please explain your answers to *part 1a* in detail.

b Assume a 1MByte direct mapped cache with cache lines of 512 bytes, and a 64 bit address space. How many bits are needed to address a word in a cache line? How are the remaining address bits used in a cache access?

c The following programme runs on a machine with a single cache as described in *part 1b*.

i. Explain how many main memory accesses (memory bus transactions) the programme below generates.

ii. Explain how much data gets transfered back and forth between memory and the cache.

You may assume:

- The cache is empty in the beginning.
- Reasonable register allocation
- Array a and array b start at the beginning of a cache line.
- Array a and array b do not interfere in the cache, and
- A "write back" policy for writes to the cache.

State any other assumptions you make.

```
int i,a[1000000],b[1000000];
init(a); b[0]=0;
for(i=1;i<1000000;i++){
    b[i]=a[i]*a[i-1]+b[i-1];
}
```

d Assuming that a cache line can be retrieved and stored in main memory in 500 clock cycles, multiplies take 3 clock cycles and additions take 1 clock cycle: How many clock cycle does the processor have to wait due to an access to main memory?

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

2a Virtual Memory

The processor issues the following instruction: `mov (R1), R2` meaning: load R2 with the value stored at the address *A* stored in `R1`. Describe in detail the process that starts with address *A*, involving a single cache and a single TLB, and ending with the data item arriving at the register `R2`. Make sure to explain the use of virtual and physical addresses, the cache, TLB, and all other structures used in the data access process.

b On many processors we can choose between a few pre-set virtual page sizes. Suggest three reasons why we would need different page sizes. Explain your three choices in detail.

c Assume a 32 bit address space, virtual page size of 4 KBytes, a paging file size of 10GBytes, and a physical main memory of 1.5 GBytes. The code below shows a matrix multiply using double precision floating point arithmetic.

```
for(i=0;i<1000000000;i++){ //result rows
    for(j=0;j<1000000000;j++){ // result columns
        for(k=0;k<1000000000;k++){
            result[i][j]=matrix1[i][k]*matrix2[k][j];
        }
    }
}
```

i. Explain how you would change the code above to improve performance?

ii. Explain how the layout of the data could be changed to obtain higher performance?

The three parts carry, respectively, 30%, 30%, 40% of the marks.

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

3a Describe the advantages of compiling programs directly into hardware.

b Describe, with the use of a circuit diagram, how the parallel execution statement:

```
PAR
    P
    Q
    R
```

can be implemented in hardware using the token-passing method.

c A `DO ... UNTIL(E)` loop differs from a `WHILE(E)` loop in that the first iteration of the loop is always executed, the expression `E` is only evaluated at the end of the loop and the loop terminates when `E` is true.

i) Provide a circuit diagram showing how a `DO ... UNTIL(E)` loop can be implemented in token-passing hardware.

ii) Hence, provide the circuit diagram for the following program implemented in token-passing hardware:

```
int_6 X
SEQ
    X := 10
    DO
        X := X - 1
    UNTIL (X = 0)
```

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

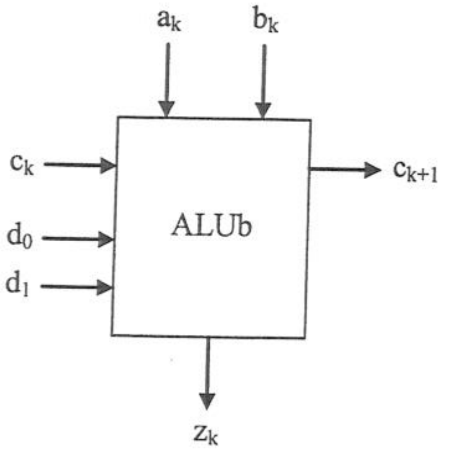
4a The executable code for program P has N instructions. A fraction α of the instructions are type Y, and the rest are type X.

i) Machine M1 has a single-cycle data-path and operates at f_1 cycles per second (Hz). How long will M1 take to execute P?

ii) Machine M2 has a multi-cycle data-path, and operates at f_2 Hz. Given that each instruction takes x cycles to execute, how long will it take M2 to execute P?

iii) Machine M3 also has a multi-cycle data-path, and operates at f_3 Hz. However, although the execution time for type X instructions is the same as for M2 (x cycles), type Y instructions are faster, taking y cycles to execute. Given that $f_2 > f_3$ determine how many times faster Y instructions need to be executed for M3 to be faster than M2 when running P.

b The circuit ALUb shown below implements one bit of a simple ALU.



d_1	d_0	c_k	z_k	c_{k+1}
0	X	0	$a_k \oplus b_k$	$a_k \cdot b_k$
0	X	1	$\overline{a_k \oplus b_k}$	$a_k + b_k$
1	0	X	$a_k \cdot b_k$	X
1	1	X	$a_k + b_k$	X

i) Provide a circuit diagram to show how ALUb can be built using a fulladder, multiplexors and two-input logic gates. Do not show the internal logic of the fulladder or the multiplexors.

ii) Provide a circuit diagram to show how four copies of the ALUb circuit can be used to build a four-bit ALU which implements add, AND and OR functions when d_1d_0 = 01, 10, 11 respectively (ignore overflow). Label all inputs and outputs.

iii) By adding logic to the carry input show how the ALUb circuit can be used to implement ALUb', which implements all the functions of ALUb as well as an exclusive-OR function on the inputs (a_k, b_k) when d_1d_0 = 00.

The two parts carry, respectively, 50%, 50% of the marks.

E2.13 Computer Architecture

Department of Computing Examinations — 2006–2007 Session *Confidential*

MODEL ANSWER and MARKING SCHEME

First Examiner oskar Paper Code C210=E2.13

Second Examiner nps Question 1 Page 1 *out of* 9

Question labels in left margin Mark allocations in right margin

1a	Exploring spatial locality of data accesses Increasing cache line size increases granularity of DRAM access. For a given system, increasing the cache line size increases conflict misses. Reducing cache line size increases initial misses and makes memory accesses less efficient	4
b	assuming 32 bit words ⇒ 128 words/line ⇒ 7 addr. bits. Remaining bits are used to check the tag.	4
c	1 cache line = 1 memory transaction 4MBytes = 8192 cache lines 3× 8192 = 24576 memory bus transactions total transfer is 12 MBytes.	6
d	4 cc for multiply add. (cc = clock cycle) For 2 cache lines (1 write, 1 read) ⇒ 1000 cc 512 bytes = 128 words 128 × 4cc = 512 cc ⇒ 488 cc wait ⇒ 487,999,512 cc total ⇒ 51,2% efficiency	6

Department of Computing Examinations — 2006–2007 Session *Confidential*

MODEL ANSWER and MARKING SCHEME

First Examiner ostkar Paper Code C210 = E2.13

Second Examiner nps Question 2 Page 2 *out of* 9

Question labels in left margin Mark allocations in right margin

2 a (R1) is a virtual address A. Address A goes to the TLB (and in case of a miss, to the OS translation table) resulting in a physical address, $\bar{A}$. $\bar{A}$ goes to the cache and in case of a miss, all the way to the main memory. In case the page is not in memory, we also have to wait for it to arrive from the paging file 6

b page size = granularity of virtual memory access 6

Different page sizes allow us to adapt to

- different storage characteristics for the paging file
- adapt to the applications data access requirement
- address a larger address space (e.g. on x86)

c Blocking improves performance, i.e. partition the matrices into smaller blocks which fit into the cache and work your way through each block separately. Layout can be improved by laying out matrix 1 in row order and matrix 2 in column order. 8

Department of Computing Examinations — 2006–2007 Session *Confidential*

MODEL ANSWER and MARKING SCHEME

First Examiner nps Paper Code C210 = E2.13

Second Examiner oskar Question 3 Page 3 *out of* 9

Question labels in left margin Mark allocations in right margin

3a Programs compiled directly into hardware can be executed in parallel, which makes them fast. 4

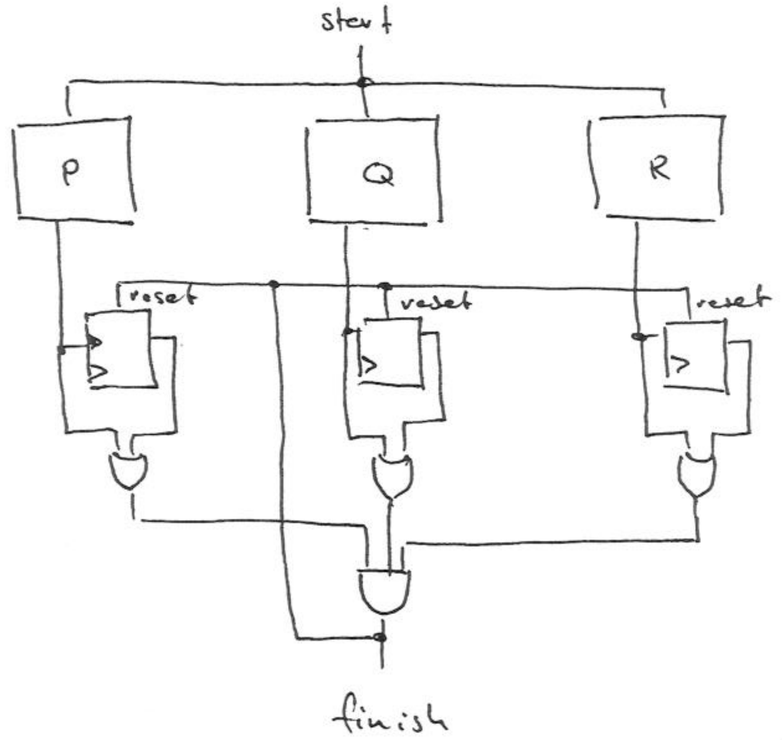
One can create processors with customised instructions, which use hardware more efficiently than general purpose processors.

MODEL ANSWER and MARKING SCHEME

First Examiner	nps	Paper Code	C210 = E2.13
Second Examiner	oskar	Question	3 Page 4 *out of* 9

Question labels in left margin | Mark allocations in right margin

3b 6



The token is split and passed to all three processes (P,Q,R).

Registers latch the tokens as each process finishes.

The AND gate makes sure the finish token is not generated until all three processes have finished.

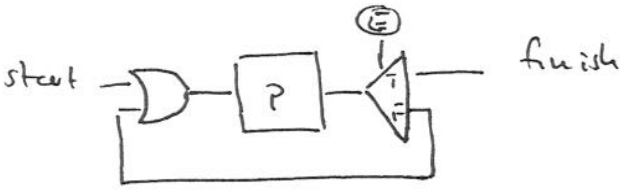
The finish token resets the registers to the initial state.

The OR gates are needed to avoid waiting an extra cycle after the slowest process completes before generating the finish token.

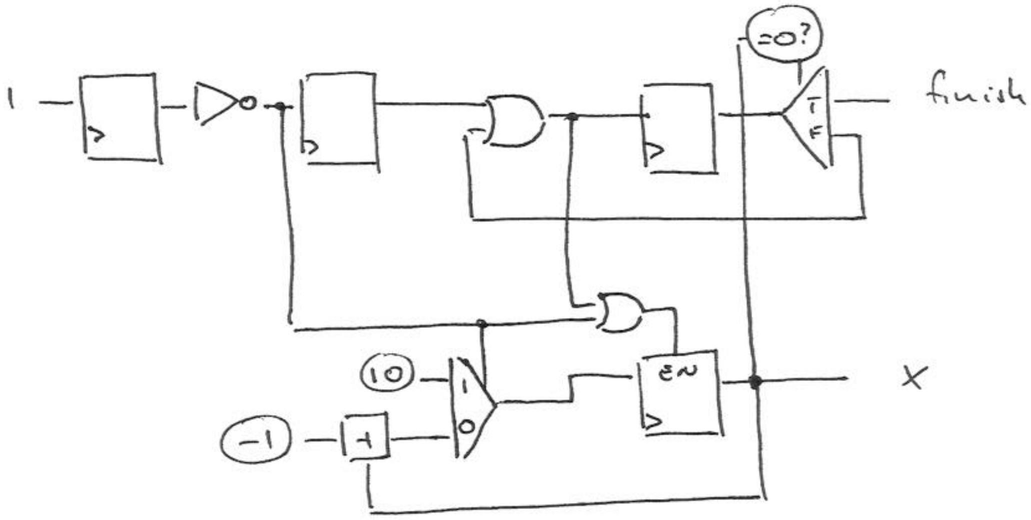
Department of Computing Examinations — 2006–2007 Session	***Confidential***
MODEL ANSWER and MARKING SCHEME	
First Examiner nps	Paper Code C210 = E2.13
Second Examiner oskar	Question 3 Page 5 *out of* 9
Question labels in left margin	Mark allocations in right margin

3c 10

i)



ii)



Department of Computing Examinations — 2006–2007 Session *Confidential*

MODEL ANSWER and MARKING SCHEME

First Examiner nps	Paper Code C210 = E2.13
Second Examiner oskar	Question 4 Page 6 *out of* 9
Question labels in left margin	Mark allocations in right margin

4a i) m1 Execution time $= \text{CPI} \times (\text{num. instructions}) \times \frac{1}{\text{clock rate}}$ 10

$$= 1 \times N \times \frac{1}{f_1}$$

$$= \frac{N}{f_1}$$

ii) m2 Execution time $= x \times N \times \frac{1}{f_2}$

$$= \frac{xN}{f_2}$$

iii) m3 execution time $= (\text{average CPI}) \times N \times \frac{1}{f_3}$

average CPI $= (1-\alpha)x + \alpha y$

$\therefore$ execution time $= [(1-\alpha)x + \alpha y]\frac{N}{f_3}$

For this to be faster than m2:

$$[(1-\alpha)x + \alpha y]\frac{N}{f_3} < \frac{xN}{f_2}$$

4 a iii) cont.

$$\alpha y < \frac{x f_3}{f_2} - (1-\alpha) x$$

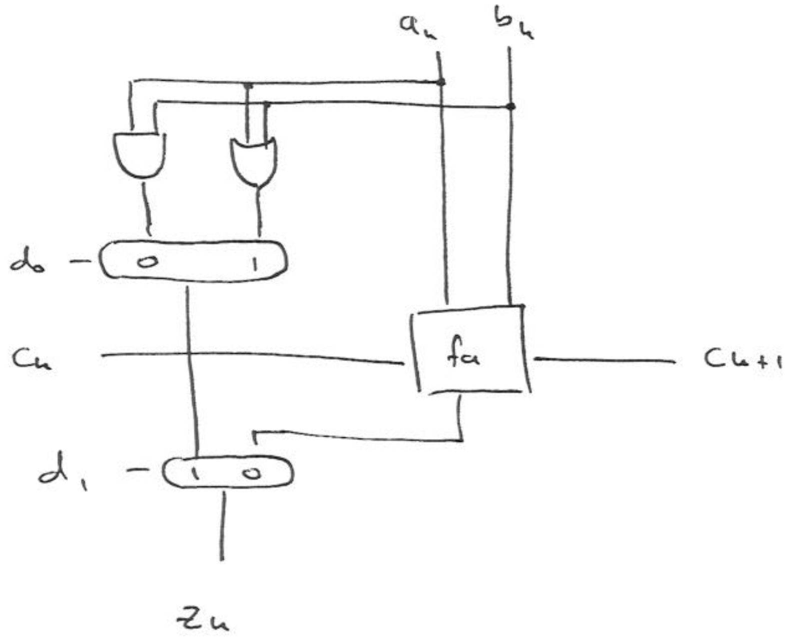
$$\frac{y}{x} < \frac{f_3/f_2 - 1 + \alpha}{\alpha}$$

$$\frac{x}{y} > \frac{\alpha}{f_3/f_2 - 1 + \alpha}$$

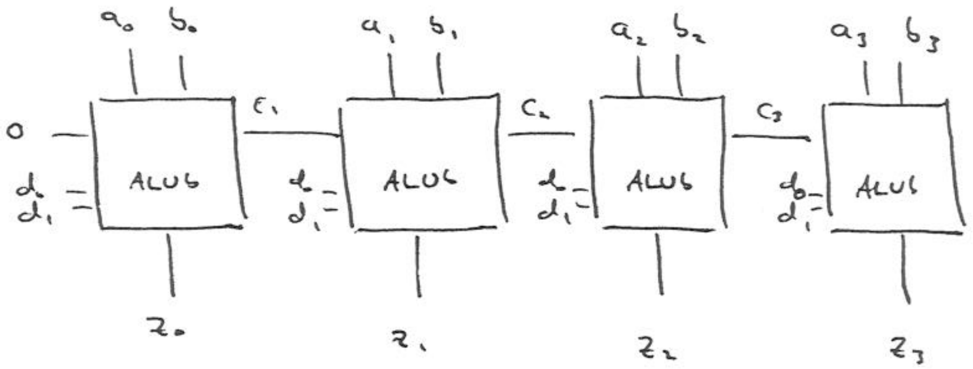
The R.H.S. is how many times faster y has to be for M3 to be quicker, at executing P, than M2.

4b 10

i)


a_n
b_n
d_0
0
1
c_n
fa
c_{n+1}
d_1
1
0
z_n

ii)


a_0 b_0
a_1 b_1
a_2 b_2
a_3 b_3
0
c_1
c_2
c_3
d_0 d_1
ALUb
d_0 d_1
ALUb
d_0 d_1
ALUb
d_0 d_1
ALUb
z_0
z_1
z_2
z_3

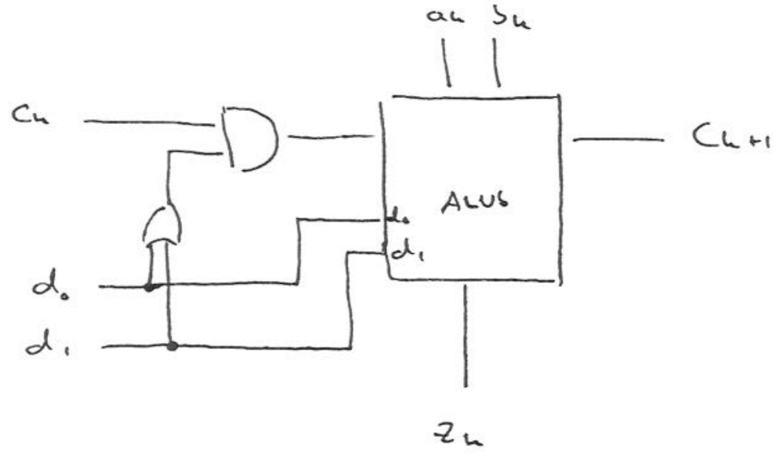
Department of Computing Examinations — 2006–2007 Session *Confidential*

MODEL ANSWER and MARKING SCHEME

First Examiner	nps	Paper Code	C210 = E2.13
Second Examiner	oskar	Question 4 Page 9 *out of* 9	
Question labels in left margin		Mark allocations in right margin	

4b

iii)


an
bn
cn
cn+1
ALUS
d0
d1
d0
d1
zn