IMPERIAL COLLEGE LONDON

TIMED REMOTE ASSESSMENTS 2021-2022

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

This paper is also taken for the relevant assessments for the
Associateship of the City and Guilds of London Institute

PAPER COMP40005

INTRODUCTION TO COMPUTER ARCHITECTURE

Monday 9 May 2022, 10:00
Writing time: 80 minutes
Upload time: 25 minutes

Answer ALL TWO questions
Open book assessment

This time-limited remote assessment has been designed to be open book. You may use resources which have been identified by the examiner to complete the assessment and are included in the instructions for the examination. You must not use any additional resources when completing this assessment.

The use of the work of another student, past or present, constitutes plagiarism. Giving your work to another student to use constitutes an offence. Collusion is a form of plagiarism and will be treated in a similar manner. This is an individual assessment and thus should be completed solely by you. The College will investigate all instances where an examination or assessment offence is reported or suspected, using plagiarism software, vivas and other tools, and apply appropriate penalties to students. In all examinations we will analyse exam performance against previous performance and against data from previous years and use an evidence-based approach to maintain a fair and robust examination. As with all exams, the best strategy is to read the question carefully and answer as fully as possible, taking account of the time and number of marks available.

Paper contains 2 questions

1 A program is designed to multiply a collection of numbers by a constant C. The RISC processor executing this program has a register file. Its multiply instruction, **`mult Reg1 Reg2 Reg3`**, takes 4 cycles to get inputs from two registers **`Reg2`** and **`Reg3`** and stores the product in register **`Reg1`**. Its add and subtract instructions, **`add Reg1 Reg2 Reg3`** and **`sub Reg1 Reg2 Reg3`**, follow the same form as the multiply instruction, but each only takes 2 cycles. The shift left instruction, **`shl Reg1 Reg2 s`**, takes one cycle to store in **`Reg1`** the value from **`Reg2`** which is left shifted by **`s`** bits, with zeros introduced from the least significant bit.

a For the constant C=7, show how the multiplication can be implemented using only the subtract and shift left instructions, with the input value *n* in register R1 and the output value *7n* in register R2.

How many cycles would this implementation take, and how many times would this implementation be faster or slower than using the multiply instruction?

b Repeat part a for the constant C=1020, using only the add and shift left instructions.

c Show how multiplication by C=1020 can be implemented based on Booth's Algorithm. Any instruction except the multiply instruction can be used. How many cycles would this implementation take, and how many times would this implementation be faster or slower than using the multiply instruction?

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

2 a *Assembly*

i) *Pointer arithmetic.* Supose x_N the address of integer array N, and integer index *i* are stored in registers `%rbx` and `%rdx`, respectively. The result is stored in register `%rax` if it is a pointer and register element `%eax` if it has data type `int`.

Following the example fill the rest of the table showing the Type, the Expression, and the Value per each assembly code line:

Assembly code	Type	Expression	Value
`movl 8(%rbx), %eax`	int	N[2]	M[x_N +8]
`leaq 8(%rbx,%rdx,4), %rax`			
`movl 12(%rbx,%rdx,8),%eax`			
`leaq 20(%rbx, %rdx,4), %rax`			

ii) *Stack discipline* Consider the following x86-64 optimised assembly function:

```
subq $24, %rsp
movw %di, 12(%rsp)
testw %di, %di
jle .L5
movl %edi, %eax
testw %si, %si
jle .L2
cmpw %si, %di
jg .L3
movswl %si, %esi
leaq 12(%rsp), %rdi
movl $0, %eax
call increment
jmp .L4
.L3:
movswl %di, %edi
movl %esi, %ecx
sall %cl, %edi
movw %di, 12(%rsp)
.L4:
movzwl 12(%rsp), %eax
jmp .L2
.L5:
movl $0, %eax
```

```
.L2:
addq $24, %rsp
ret
```

Fill in the blanks in the following `loop` function. You may only use the variable names `x` and `y`, not register names.

```
short compute(short x, short y){
    if( ________)
      return ________;
    if(_________)
      return ________;
    if(_________)
        ________;
    else
        ________________;

    return _____________;
}
```

b *Memory Hierarchy.* A certain memory system has a 128MB main memory and a 2MB cache. Blocks are 32 Bytes in size. Show the fields in a memory address if the cache is:

a associative

b direct-mapped

c 8-way set-associative

If the following sequence of addresses is sent to the cache, **0, 32, 8, 32, 64**. What is the miss rate? Show your working.

The two parts carry, respectively, 65% and 35% of the marks.