You can find old Computer-Science exam papers via:
You have also been given lots of possible or old exam part-questions in the handouts, and you can ask about
any of them during these examples classes.
The format of the January COMP15111 exam will be very similar to that for previous COMP10031 and
COMP15111 exams:
− You have One and a half hours
− Answer Question 1 (set by both Pete and Richard)
− Answer either Question 2 (set by Pete) or Question 3 (set by Richard)

Question 1 consists of many parts, each worth about 2 to 4 marks.
Questions 2 and 3 consist of a few parts, each worth about 5 to 10 marks.
**Warning:** the things we ask about in Questions 2 and 3 will be similar to previous years, but
the topics asked about in each question may change, as the split of topics between lecturers has
changed.
The exams for January 2005 and following years were set to the same syllabus, so should give
you some idea what to expect.

## 1 January 2005 exam

**Question 1. This question is compulsory.**

a) Explain the concept of a Pseudo Instruction in ARM assembly language. (2 marks)
b) Describe the ARM ADRL pseudo instruction and discuss the circumstances in which it must generate more
than one real instruction. (4 marks)
c) What is bytecode in the context of Java and how does it get executed by a real processor? (4 marks)
d) What is the 3-address instruction style? What is the load-store instruction style? Give code sequences for
"a = b + c + d" in both styles. (2 marks)
e) What is a bit? How is a bit used to represent a simple decision, like the answer to "Is this exam hard"? What
is a byte? How might a byte be used to represent a character? What is a word? How is a word used to
represent a number or an address? (3 marks)
f) Explain how to convert (decimal) 97 to binary, and then to octal and to hexadecimal. (2 marks)
g) In the context of a computer program accessing a peripheral, for each of the following, explain what it means
and give an example of a situation in which it would make sense to use it:
\( i) \) polling \( ii) \) an interrupt \( iii) \) direct memory access (DMA) (3 marks)

**Question 2.**

a) Explain the difference between direct and indirect addressing. (2 marks)
b) Explain base plus offset addressing as implemented in the ARM processor. (2 marks)
c) Describe PC relative addressing and discuss why it is of limited use to access large amounts of data in an
ARM program. Explain how more general base plus offset addressing can be used to solve this problem. (4 marks)
d) Explain the concept of auto indexing with the aid of an example. (4 marks)
e) Discuss the circumstances in which the Link Register (LR) in the ARM processor can be used in a subroutine
call without the aid of a stack. You should include an explanation of why stacking LR is necessary in many
circumstances. (4 marks)
f) A subroutine is passed two integer parameters which it adds together and returns the result in R0. Assuming
that the subroutine places the Link Register on the stack and uses the stack to pass the parameters, sketch the
ARM code required for a simple implementation of the subroutine and draw diagrams of the resulting stack
structure. (4 marks)

**Question 3.**
a) What is meant by a "stored program" computer? What is the fetch-execute cycle? Describe in detail what happens when the following ARM program is run. Clearly describe the movement of information between the memory and the CPU, and how the values in the registers (including PC) and memory change, at each step. Assume that the program starts at memory location 0. (6 marks)

```assembly
B start
one DEFW 23
two DEFW 45
three DEFW 10
start LDR R0, one
    STR R0, three
    LDR R1, two
    ADD R2, R1, R0
    STR R2, one
    SWI 2; terminate program
```

b) What are the differences between an assembler and a compiler? Outline the main 4 phases that they have in common. (6 marks)

c) You are required to translate the following Java statements into an equivalent sequence of ARM instructions (a and b are integer variables).

```java
while (a - b > 0) {
    if (a < -b) {
        b = b - a;
        a = -a;
    } else {
        b = a * b;
        a = 2 - b;
    }
}
```

i) Give the ARM code assuming that the variables can be easily accessed (e.g. using short offsets from the PC register) and that you must reload them into registers each time they are used.

ii) There are some optimisations that you can make e.g. keeping variables in registers, or using special instructions available on the ARM. For each optimisation that you can apply to this example, illustrate it by explaining the changes that would have to be made to your original answer. In particular, you should discuss the use of conditional arithmetic instructions, and whether they would be an improvement in this example. (8 marks)

2 Some very old exam questions

To help you revise, we have also looked back at some old exams to pick out parts of the questions that you ought to be able to answer – some have been slightly modified, to make them suitable.

Question 1 (compulsory)

Note: an exam question is almost always marked out of 20. However, the marks for these part-questions add up to about 30, so all of this section is 50% longer than one typical exam question.

(a) Express the 8-bit binary pattern 0101 1010
   (i) in hexadecimal notation (ii) in octal notation (2 marks)

(b) To which decimal number does the 8-bit binary pattern 1001 0011 correspond, assuming that it represents an unsigned binary integer. (1 mark)

(c) What are the results (in hexadecimal form) of the following logical operations
   (i) 0x7e AND 0x21 (ii) 0x7e XOR 0x21 (XOR is called EOR on the ARM) (2 marks)

(d) Explain the effect of the following ARM instruction on the registers involved and on memory:
   STMFD SP!, {R0, R3-R5} (2? marks)

(e) How many address bits are required to address 32 Mbyte of byte-addressable memory? (2 marks)
(f) Explain the difference between the instruction: \texttt{ADD R0, R0, R0} and the instruction: \texttt{ADDS R0, R0, R0} (2 marks)

(g) If register R1 contains the value 0x5c and register R2 contains the value 0x6a, what are the results of the following ARM instructions:

(i) \texttt{EOR R0, R1, R2}  (ii) \texttt{AND R0, R1, R2} (4 marks)

(h) Explain the purpose of the CPSR register in an ARM processor (2 marks)

(i) Express the decimal number 101 as an unsigned 8-bit binary number. Give your answer in hexadecimal form. (1 mark)

(j) Assuming that the registers R1, R2, R3, R4 contain the values 0x11aa, 0x22bb, 0x33cc, 0x44dd respectively, and that the register R5 contains the value 0x1000, what is the value in R5 after each of the following ARM instructions in this program fragment and what byte value is stored in each affected memory location? (Assume a little-endian configuration.) (4 marks)

\begin{verbatim}
STR R1, [R5]
STR R2, [R5, #4]
STR R3, [R5], #4
STR R4, [R5, #4]!
\end{verbatim}

(k) A laser printer contains an ARM processor with two types of memory, ROM and RAM. The ROM starts at address 0x00000000 and is 8 Mbytes in size. The RAM is 64Mbytes in size and is located immediately above the ROM in the memory map. What is the address of the lowest RAM word address? What is the address of the highest RAM word address? (2 marks)

(l) The format of an ARM data processing instruction is:

\begin{center}
\begin{tabular}{cccccccc}
\textbf{Cond} & 2 & 1 & 4 & 1 & 4 & 4 & 12 \\
\hline
\texttt{00} & \# & \texttt{OpCode} & S & Rn & Rd & Operand 2 \\
\hline
\end{tabular}
\end{center}

The “#” bit determines the form of Operand 2.
The “S” bit determines whether the condition codes are set (S=1) or not (S=0).

The instruction 0xe04ec00f represents one particular form of the SUB instruction.

(i) What is the destination register Rd?
(ii) What is the first source register Rn?
(iii) What 4-bit OpCode represents a SUB instruction?
(iv) Are the condition codes updated by this instruction? (4 marks)

The following are optional questions from old exams. I have rewritten or deleted some parts, and added [notes]. Even so, some of these questions cover things that have been mentioned, rather than explored in detail, and so would be unfair to ask in this year’s exam.

January 2002 Question 2

a) Register R0 is multiplied by what number as a result of the following 2 instructions executed in sequence? (4 marks)

\begin{verbatim}
ADD R0, R0, R0,LSL #3
RSB R0, R0, R0,LSL #2
\end{verbatim}

b) Only certain literal values can be used in ARM instructions:

i) Explain why this is so. (2 marks)

ii) Explain how a range of literal values can be generated. (2 marks)

iii) What literal is moved to R2 by the following instruction? \texttt{MVN R2, #0xff,6} (2 marks)

c) The following code is an implementation of Euclid’s GCD algorithm. [This calculates the Greatest Common Divisor – also known as the Highest Common Factor – of two numbers e.g. 6 and 9 would give 3.]

\begin{verbatim}
gcd CMP R0, R1
BEQ end
BLT less
SUB R0, R0, R1
B gcd
\end{verbatim}
less SUB R1, R1, R0
    B gcd
end

i) Rewrite the code using conditionally executed instructions. (4 marks)
ii) What are the benefits of ARM’s ability to conditionally execute all instructions? (2 marks)
iii) Quantify the benefits for your answer for the example above. (4 marks)

[For part (a) the operand “R0,LSL #3” means use $R0 \times 2^3$, rather than just $R0$, as the value. For part (b) the final “6” refers to how the literal is shifted within a 32 bit value – guess what happens!]

January 2003 Question 2

a) All instructions in the ARM instruction set may be conditionally executed. Discuss the advantages and disadvantages of this feature. (4 marks)
b) The following subroutine prints in hexadecimal format a number held in register R1. Explain the purpose of each line of code.

Note: it is not enough to merely state the effect of a particular instruction, you should also indicate how that statement contributes to the goal of the subroutine. For example, the correct explanation of the statement “HexOut MOV R2, #8” is not “move the value 8 into r0”, but rather "HexOut is a label which identifies the subroutine. Register R2 is used as a loop counter and is initialised to the value 8 because there are 8 hexadecimal digits to be printed" (10 marks)

HexOut MOV R2, #8
loop MOV R0, R1,LSR #28
    CMP R0, #9
    BHI ascii
    ADD R0, R0, #"0"
    SWI 0
    B next
ascii ADD R0, R0, #"A"-10
    SWI 0
next MOV R1, R1,LSL #4
    SUBS R2, R2, #1
    BNE loop
    MOV PC, LR

c) Rewrite this code taking advantage of conditionally executed instructions to remove unnecessary branches. (6 marks)

January 2004 Question 2

a) What is meant by a stack? (2 marks)
b) Explain the “FD” stack organisation. (2 marks)
i.e. how does SP change in value after a pop or push, and where is SP pointing relative to the last thing pushed?]
c) Show how the STR and LDR instructions can be used to push values to and pop values from the stack. (2 marks)
d) Discuss the differences between LDR/STR instructions and LDM/STM instructions. (4 marks)
f) A non-leaf method requires the use of registers R0, R1, R2 which are currently in use by the code calling the method. What code is necessary at entry to and exit from the method to preserve all registers in use? (4 marks)

This is not a complete question. For part (f) a “leaf” method is one that calls other methods, and a “non-leaf” method doesn’t.

January 2004 Question 3

a) What exception (interrupt) types are supported by an ARM processor. Briefly explain each one. (6 marks)
b) Describe how exceptions are serviced in an ARM processor. (4 marks)
c) Why does a processor need privileged modes? (2 marks)
d) What extra registers are used by the privileged modes on the ARM? (4 marks)

e) Explain how the ARM Fast Peripheral Interrupt mode has been designed to minimise the response time in servicing external interrupts. (4 marks)

[This is not a complete question, and we don’t really expect you to have memorised all the different exceptions used on the ARM for part (a)]