COMP15111

One and a half hours

"ARM Instruction Set Summary" is attached

Questions A1 and B1 are COMPULSORY

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Fundamentals of Computer Architecture

Date: Monday 25th January 2016

Time: 14:00 - 15:30

Please answer Questions A1 and B1
and
ONE other Question: either A2 or B2

Use a SEPARATE answerbook for each SECTION

This is a CLOSED book examination

The use of electronic calculators is NOT permitted

[PTO]

Section A

Compulsory

A1. Computer Architecture

a) Translate the decimal number 347_{10} to hexadecimal and the octal number 135_8 to decimal **without using binary** as an intermediate representation. (4 marks)

b) Explain **briefly** the following addressing modes and their pros and cons. (3 marks)

i) Direct addressing
ii) Base+Offset addressing
iii) Post-index addressing

c) Explain the differences between 3-address style instructions, 1-address style instructions and load-store style instructions. (3 marks)

Optional

A2. ARM

a) Explain the similarities and differences between the following pairs of ARM mnemonics: (4 marks)

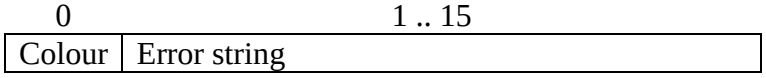
i) STR and STMFD
ii) SUB and SUBS
iii) ADD and ADDEQ
iv) ADR and ADRL

b) Consider the following switch statement to manage a keyboard-controlled menu in an embedded system:

```
switch (option) {
  case 'd': delete();
            break;
  case 'm': move();
            break;
  case 'r': rename();
            break;
  default:  invalid_option();
            break;
}
```

Describe, without giving any code, two different ways of implementing it in ARM. Which one of them would you choose if **reducing memory footprint** was the main concern of the target embedded system: (6 marks)

c) You need to implement an error reporting method which, given an error code as a parameter (in R0) it looks up in a table for the corresponding printing colour and error string. This table (labelled `error_table`) has 20 elements with a fixed width of 16 bytes. In each element the least significant byte is used to store a colour code and the other 15 bytes store an error string (see below). In order to print the error code in the required colour in a terminal you need to used the method `print_colour` which accepts 2 parameters: the colour to be used in R0 and the address of the first character to print in R1. Your function needs to return a completion code in R0: 1 if the printing was successful or -1 if there was any error.



Make sure your code is **safe** and try to **optimise** it as much as you can, explaining what optimizations you do. (10 marks)

Section B

Compulsory

B1. Java Virtual Machine

a) With respect to the 'stack' in a Java Virtual Machine? Brief explain the types of information the 'stack' holds. (2 mark)

b) With respect to the JVM and in relation to the "3-steps taken to create a new object" briefly explain each step; draw up a set of 3-diagrams (as outlined in Figure B1.b.), fully label and annotate each, given this Java code:

```
String  s;
s = new String("abc");
```

and the address it is allocated is 1000. (4 marks)

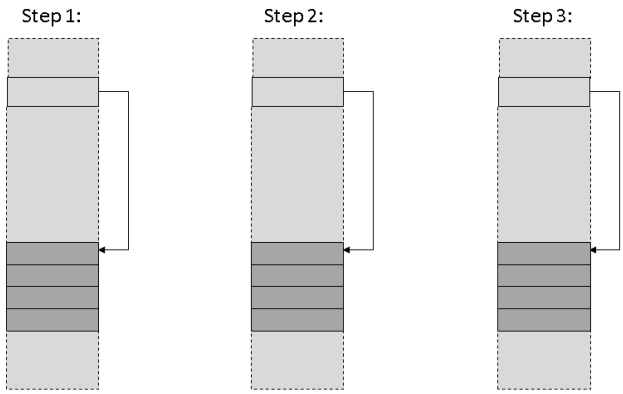


Figure B1.b., a diagram explaining the 3-steps to create a new object.

c) An Operating System Kernel consists of basically five main functions; name the five main functions a Kernel performs for an OS and briefly describe each. (4 marks)

Optional

B2. Java Virtual Machine and Peripherals

a) In your lectures you were introduced to code, in the form of a loop for 'polling a simple device;' which would be something like that presented in Figure B2.a. State [explicitly] the function each line performs in the comment column; when you copy Figure B2.a. into your answer book.

Hint: this means you should be as concise and explicit in the explanation of the: labels, instructions [operation], operand(s), registers, variables etc.; as possible.

(4 marks)

Label	Operation	Operand	Comments
loop	ADR	R1, status_reg	
	LDRB	R0, [R1]	
	TST	R0, #0x80	
	BEQ	loop	
	ADR	R1, data_reg	
	LDRB	R0, [R1]	

Figure B2.a., A table showing code for a loop that can check a peripheral.

b) In the context of data exchange between CPU and peripherals. Differentiate between the two main data exchange protocols; polling and interrupts (their different steps/phases). (4 marks)

c) Explain what is meant by the term Java Bytecode? (4 marks)

d) Explain how an expression of the form x = (a-b)*(c+d) is evaluated using [a sequence of] bytecode instructions. Typical bytecode instructions are: PUSH and POP. (4 mark)

e) In general array access involves a computed index which is added to the base of the array. Draw up a diagram that depicts 'a' the base, 'i' the index. The array has ten elements of 32 bits each. Depict the case where 'i=5' in your diagram; diagram should be fully labelled and annotated. (4 marks)

END OF EXAMINATION

COMP15111 ARM Instruction Set Summary

This is not a complete list and you may need to consult other documentation for more detail.

Data Processing

ADD	Rd, Rn, Op	Rd= Rn + Op
SUB	Rd, Rn, Op	Rd= Rn – Op
RSB	Rd, Rn, Op	Rd= Op – Rn ("reverse subtract")
AND	Rd, Rn, Op	Rd= Rn AND Op
ORR	Rd, Rn, Op	Logical OR
EOR	Rd, Rn, Op	Exclusive OR
BIC	Rd, Rn, Op	Bit Clear: Rd= Rn & !Op
MOV	Rd, Op	Rd= Op
MVN	Rd, Op	Rd= !Op
CMP	Rn, Op	set status on Rn – Op
CMN	Rn, Op	set status on Rn + Op
TST	Rn, Op	set status on Rn AND Op
TEQ	Rn, Op	set status on Rn EOR Op
MUL	Rd, Rn, Rs	Rd= Rn * Rs
MLA	Rd, Rn, Rs, Rp	Rd= (Rn * Rs) + Rp

An "Op" in the table above can be a literal (e.g. `#10`) or a register (e.g. `R1`) or a shifted register.

If a shift (by a literal e.g. `#3` or a register e.g. `R6`) is required it is specified as the fourth parameter e.g.
`ADD R1, R4, R5, LSL #3` ; R1 = R4 + R5*8
`ADD R1, R4, R5, LSL R6` ; R1 = R4 + R5*2^{R6}

LSL Shift	logical shift left by $0 \leq Shift \leq 31$ places, putting 0s into least significant end
LSR Shift	logical shift right by $0 \leq Shift \leq 32$ places, putting 0s in most significant end
ASR Shift	arithmetic shift right by $0 \leq Shift \leq 32$ places, copying the sign bit into the most significant end
ROR Shift	circular rotate right by $0 \leq Shift \leq 32$ places, moving bits lost from one end into other end
RRX	one place right shift. Carry from status reg. shifts into most significant bit. If status reg. set by instruction, carry bit = least significant bit. This gives a 1-bit circular rotate through the carry bit

Loads and Stores

LDR	Rd, Address	loads a 32-bit word into Rd from memory location
LDRB	Rd, Address	loads 8-bits into Rd; top 24 bits are '0's
STR	Rd, Address	stores Rd at memory location
STRB	Rd, Address	stores bottom byte of Rd at memory location
LDMFD	Rd, register list	multiple reg load,load from addr Rd
STMFD	Rd!, register list	multiple reg store, Rd is updated after each store operation

An "Address" in the table above can be a numerical address (e.g. `100`) or a named memory location (e.g. `fred`) or calculated from the contents of registers and literals e.g.:

operand form	address	final value of R0
$[R0]$	R0	(unchanged)
$[R0, R1]$	R0 + R1	(unchanged)
$[R0, \#1]$	R0 + 1	(unchanged)
$[R0], R1$	R0	R0 + R1
$[R0], \#1$	R0	R0 + 1
$[R0, R1]!$	R0 + R1	R0 + R1
$[R0, \#1]!$	R0 + 1	R0 + 1

In the examples above the value copied from `R1` can be modified by a shift (but `R1` itself is unchanged) e.g.
`LDR R2, [R0, R1, LSL #3]` ; address = R0 + 8*R1
The possible shifts are LSL, LSR, ASL and ROR as described for Data Processing instructions above.

Please Turn Over

COMP15111 ARM Instruction Set Summary continued

Control Transfer

B	Label	branch to label: R15= label
BL	Method	branch and link: R14= R15, R15= method
SWI	Number	software interrupt

Condition Codes

Any instruction can be made conditional e.g. `ADD` can become `ADDEQ` etc.

Code	Meaning	Flag condition
EQ	Equal	Z
NE	Not Equal	!Z
GE	Greater than or Equal (signed)	N = V
GT	Greater Than (signed)	(N = V) . !Z
LT	Less Than (signed)	N != V
LE	Less than or Equal (signed)	(N != V) + Z
HI	Higher (unsigned)	C . !Z
LS	Lower or Same (unsigned)	!C + Z
CS/HS	Carry Set/Higher or Same (unsigned)	C
CC/LO	Carry Clear/Lower (unsigned)	!C
MI	Minus (negative)	N
PL	Plus (positive)	!N
VS	Overflow Set	V
VC	Overflow Clear	!V
AL	Always	TRUE

Also, any Data Processing instruction can be followed by an "S" e.g. `ADD` can become `ADDS` meaning that the result of the instruction is to be compared with zero.