Ch28

Low-level Programming

CHAPTER ASSIGNMENT · 8 MARKS

Name _____ Class _____ No. _____ Date _____

Answer every question in the space given. Show your working. You may use the chapter page to help you.

Questions from *Computer Science for Cambridge International AS & A Level Coursebook*, 2nd edition, Sylvia Langfield and Dave Duddell (Cambridge University Press), for classroom use.

Tasks and questions from the chapter

28.01 Write assembly code instructions for this sequence of pseudocode statements:

```
A <- 2
B <- 10
C <- A + B
D <- A - B
```

28.02 Write assembly code instructions for this sequence of pseudocode statements:

```
IF A <> 0
  THEN
    B <- A
  ELSE
    B <- B - 1
```

28.03 Write assembly code instructions for this sequence of pseudocode statements:

```
Number <- 1
Total <- 0
Max <- 5
REPEAT
    Total <- Total + Number
    Number <- Number + 1
UNTIL Number = Max
```

28.04 Write assembly code instructions for this sequence of pseudocode statements:

```
Count <- 0
REPEAT
    Count <- Count + 1
    INPUT Character
UNTIL Character = "N"
```

28.05 Modify your assembly code instructions from Task 28.04 to implement this sequence of pseudocode statements:

```
Count <- 0
REPEAT
    Count <- Count + 1
    INPUT Character
UNTIL Character = "N"
OUTPUT Count
```

28.06 Write instructions to reverse a word entered at the keyboard. This requires access to an area of memory treated as a stack.

RP **Reflection Point:** Here are the addressing modes you should be able to use when writing low-level code:

Addressing mode	Tick / Cross
Immediate	
Direct	
Indirect	
Indexed	
Relative	

Tick the box next to each addressing mode you can use easily and place a cross in the boxes next to the addressing modes you have difficulty with.

Exam-style questions

1 The instruction set of a processor with one general-purpose register, the accumulator, includes the following instructions.

Label	Op code	Operand	Explanation
	LDD	<address>	Direct addressing. Load the contents of the given address to ACC
	STO	<address>	Store the contents of ACC at the given address
	ADD	<address>	Add the contents of the given address to the ACC

Label	Op code	Operand	Explanation
	IN		Key in a character and store its ASCII value in ACC
	AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>
	LSL	#n	Bits in ACC are shifted logically n places to the left. Zeros are introduced on the right-hand end
	END		Return control to the operating system
<label>:	<data>		Gives a symbolic address <label> to the memory location with contents <data>

Key to the above table:

- `ACC` denotes the Accumulator.
- `#` denotes immediate addressing.
- `&` denotes a hexadecimal number, e.g. `&4A`.
- `<address>` can be an absolute address or a symbolic address.

(a) Explain the operation of the `AND` instruction. [1]

(b) The ASCII code for '0' is the binary value 00110000. The ASCII code for '1' is the binary value 00110001.

Write an `AND` instruction to convert any numeric digit stored in `ACC` in the form of an ASCII code to its eight-bit binary integer equivalent. [1]

(c) Write the assembly code instructions to convert a two-digit number keyed in at the keyboard to its BCD representation. Store the result in the memory location labelled `Result`. [7]

Label	Op code	Operand	Explanation
			Input first digit
			Convert from ASCII to its digit value
			Move to upper nibble
			Store in location Result
			Input second digit
			Convert from ASCII to its digit value
			Combine the two values
			Store result
			End of program
Mask:			Mask to convert from ASCII to digit equivalent
Result:	&00		Memory location for result

2 A given processor has one general-purpose register, the accumulator `ACC`, and one index register, `IX`. Part of the instruction set for this processor is as follows.

Label	Op code	Operand	Explanation
	LDM	#n	Immediate addressing. Load the number n to ACC
	LDD	<address>	Direct addressing. Load the contents of the given address to ACC
	LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC
	LDR	#n	Immediate addressing. Load the denary number n to IX
	STO	<address>	Store the contents of ACC at the given address
	STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated address
	ADD	<address>	Add the contents of the given address to the ACC
	INC	<register>	Add 1 to the contents of the register (ACC or IX)
	JMP	<address>	Jump to the given address
	CMP	<address>	Compare the contents of ACC with the contents of <address>
	CMP	#n	Compare the contents of ACC with the denary number n
	JPE	<address>	Following a compare instruction, jump to <address> if the compare was True
	JPN	<address>	Following a compare instruction, jump to <address> if the compare was False
	IN		Key in a character and store its ASCII value in ACC
	END		Return control to the operating system
<label>:	<op code>	<operand>	Labels an instruction
<label>:	<data>		Gives a symbolic address <label> to the memory location with contents <data>

Key to the above table:

- `#` denotes immediate addressing.
- `<address>` can be an absolute address or a symbolic address.

Write an assembly language program that outputs a sequence of characters stored in successive locations, starting at the location labelled: `STRING`. Output ends when the character in ACC is '!' (ASCII code 33). [8]

Label	Op code	Operand	Explanation
			Set index register to zero
			Load ACC with character stored at STRING (modified by index register)
			Increment index register

Label	Op code	Operand	Explanation
			Output character
			Is this character the ! key?
			No – jump to beginning of loop
			End of program
STRING:		72	String stored from here onwards
		69	
		76	
		80	
		33	

End of assignment · Check your answers against the chapter at gurujazzy.com/subjects/computer-science/as-a-level/low-level-programming/