Past-paper walk-through

Assembly Language ▪ 4.2

eXercise

Questions in this set

- 9618/11 N25 Q4 ▪ 4 marks
- 9618/12 N25 Q3 ▪ 10 marks
- 9618/13 N25 Q6 ▪ 11 marks
- 9618/32 N25 Q11 ▪ 7 marks
- 9618/12 J25 Q7 ▪ 8 marks
- 9618/13 J25 Q5 ▪ 9 marks
- 9618/12 N24 Q8 ▪ 11 marks
- 9618/11 J24 Q4 ▪ 7 marks

Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

Question 4

9618/11 N25 ▪ Q4 ▪ 4 marks

The table shows part of the instruction set for a processor. The processor has one register, the Accumulator (ACC).

Instruction		Explanation
Opcode	**Operand**	
AND	#n / Bn / &n	Bitwise AND operation of the contents of the ACC with the operand
AND	<address>	Bitwise AND operation of the contents of the ACC with the contents of <address>
XOR	#n / Bn / &n	Bitwise XOR operation of the contents of the ACC with the operand
XOR	<address>	Bitwise XOR operation of the contents of the ACC with the contents of <address>
OR	#n / Bn / &n	Bitwise OR operation of the contents of the ACC with the operand
OR	<address>	Bitwise OR operation of the contents of the 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.
LSR	#n	Bits in ACC are shifted logically n places to the right. Zeros are introduced on the left-hand end.

Question 4

- # denotes a denary number, e.g. #127
- B denotes a binary number, e.g. B10010001
- & denotes a hexadecimal number, e.g. &4A

(a) The ACC currently contains the following positive binary integer:

0	0	0	1	1	1	1	0

Question 4

Write a bit manipulation instruction that uses a binary shift to change the contents of the ACC to:

0	1	1	1	1	0	0	0

[1]

Question 4

Instruction		Explanation
Opcode	**Operand**	
AND	#n / Bn / &n	Bitwise AND operation of the contents of the ACC with the operand
AND	<address>	Bitwise AND operation of the contents of the ACC with the contents of <address>
XOR	#n / Bn / &n	Bitwise XOR operation of the contents of the ACC with the operand
XOR	<address>	Bitwise XOR operation of the contents of the ACC with the contents of <address>
OR	#n / Bn / &n	Bitwise OR operation of the contents of the ACC with the operand
OR	<address>	Bitwise OR operation of the contents of the 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.
LSR	#n	Bits in ACC are shifted logically n places to the right. Zeros are introduced on the left-hand end.

Question 4

		are introduced on the left-hand end.

<address> can be an absolute or symbolic address
# denotes a denary number, e.g. #127
B denotes a binary number, e.g. B10010001
& denotes a hexadecimal number, e.g. &4A

Question 4

Address	Data
98	00100100
99	00110001
100	00110011
101	10100011
102	10101100

Question 4

0	0	0	1	1	1	1	0