Chapter 15 Student Book Answers

15.1 What you should already know

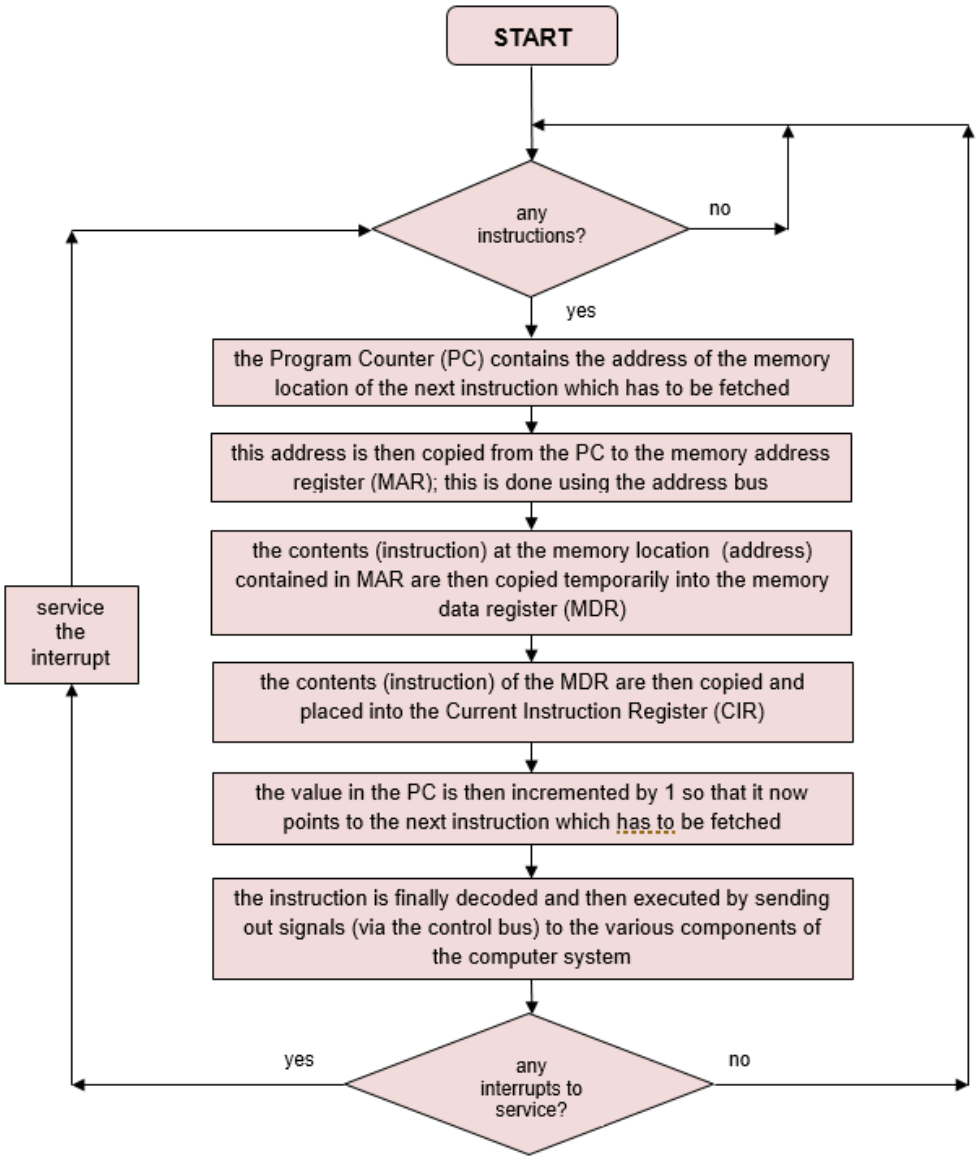
1 i)

$$\begin{array}{rl} 00111100 & C = 0 \\ +\ \underline{01000110} & V = 1 \\ \underline{10000010} & N = 1 \end{array}$$

ii)

$$\begin{array}{rl} 11000100 & C = \\ +\ \underline{10111010} & V = 1 \\ \underline{101111110} & N = 0 \end{array}$$

2



(3) (a) (i) Bus width

- word size used by the computer
- size of memory location which can be directly addressed/accessed

ii) smallest width = control bus

iii) **Address bus upgrade**

- larger word size can be used
- more addresses can be accessed directly.

b) i) **Clock**

- clock speed defines clock cycle that the computer system uses to synchronise all operations
- increasing clock speed can increase processing speed of computer.

ii) **Interrupts**

- interrupt is a signal sent to CPU by a device/program,/user which requires CPUs attention according to priority level
- interrupts can be caused by
  - i/o processing (e.g. disk drive is ready)
  - hardware fault (e.g. paper jam in printer)
  - program error (e.g. division by zero would produce software error)
  - user interaction (e.g. user presses the <BREAK> key on keyboard.

Activity 15A

1 a)

- RISC processor architecture has fewer built-in instructions which can actually lead to higher computer performance.
- RISC design strategy is built on simple instructions which is achieved by breaking up complex instructions into simpler sub-operations where each instruction requires one clock cycle.

b) **CISC features:**

- many instruction formats are possible
- there are more addressing modes
- makes use of multi-cycle instructions
- instructions can be of a variable length
- longer execution time for instructions
- decoding of instructions is more complex
- it is more difficult to make pipelining work
- the design emphasis is on the hardware
- uses the memory unit to allow complex instructions to be carried out.

RISC features:

- uses fewer instruction formats/sets
- uses fewer addressing modes
- makes use of single-cycle instructions
- instructions are of a fixed length
- faster execution time for instructions
- makes use of general multi-purpose registers
- easier to make pipelining function correctly
- the design emphasis is on the software
- processor chips require fewer transistors.

2 a) Von Neumann bottleneck:

- shared bus between program memory and data memory leads to the bottleneck …

- … so, throughput limitation due to inadequate data transfer rates between memory and CPU …
- … this causes CPU to wait and remain idle for a period of time while low speed memory is being accessed.

b) This slows down the performance of a computer system.

3 **a)** In a cluster, each machine is independent of the other computers in terms of memory and back-up store; the computers are interconnected in some variation of a network.

In massively parallel processing there is really only one machine with many thousands of CPUs/processors interconnected.

b) There are many applications in scientific and medical research (reader should pick one example form a huge list).

4 A = LOAD A

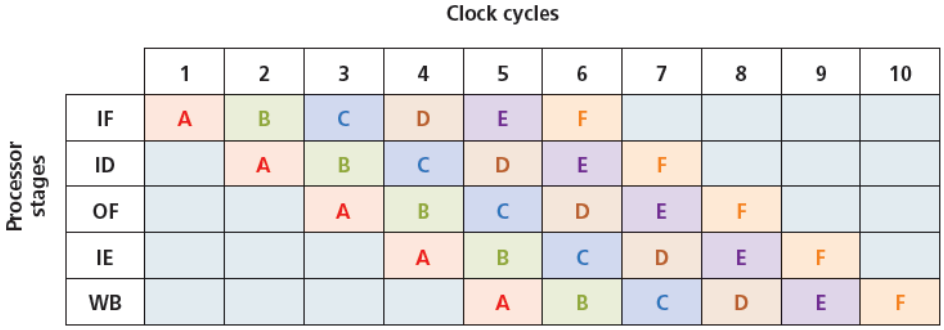
B = LOAD B

C = LOAD C

D = ADD A,B,C

E = STORE D

F = OUT D



15.2 What you should already know

1

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

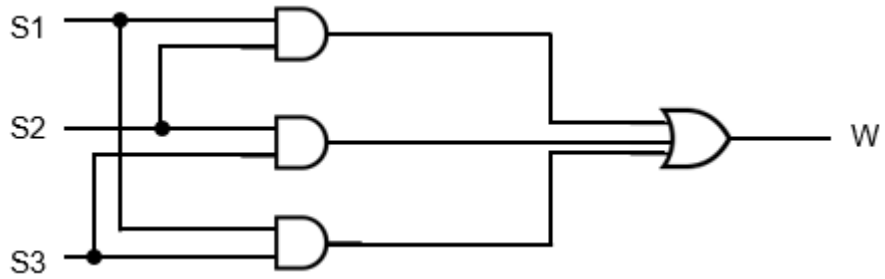
2

P	Q	R	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

((P.Q) + (Q + R)).R becomes “INPUT R” after circuit simplification

3 a) (S1.S2) + (S2.S3) + (S1.S3)

b)



c)

S1	S2	S3	W
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Activity 15B

1 a) $A.\overline{B} + A.\overline{C} + A.D + B.\overline{C}.D$

b) Let X = original expression; thus $\overline{X}$ (from a Karnaugh map) gives: $\overline{A}.\overline{B}.\overline{C}.D$

Using de Morgan's Law then gives: $\overline{A} + B + \overline{C} + D$

c)

$\overline{A}.B.C + A.B.\overline{C} + A.B.C + \overline{A}.B.\overline{C}$
$\Rightarrow B.C.(\overline{A} + A) + B.\overline{C}.(\overline{A} + A)$
$\Rightarrow B.C.1 + B.\overline{C}.1$
$\Rightarrow B.(C + \overline{C}) \Rightarrow B.1$
$\Rightarrow B$

d)

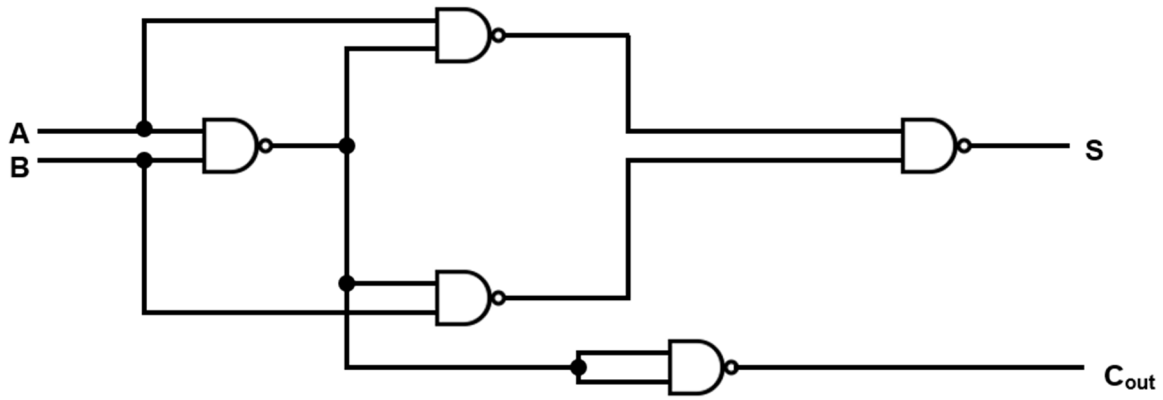
$\overline{A}.(A + B) + (B + A.A).(A + \overline{B})$
$\Rightarrow \overline{A}.A + \overline{A}.B + (B + A).A + (B + A).\overline{B}$
$\Rightarrow \overline{A}.B + (B + A).A + (B + A).\overline{B}$
$\Rightarrow \overline{A}.B + B.A + A.A + B.\overline{B} + A.\overline{B}$
$\Rightarrow \overline{A}.B + B.A + A + A.\overline{B}$
$\Rightarrow \overline{A}.B + A.B + A.1 + A.\overline{B}$
$\Rightarrow \overline{A}.B + A.(B + 1 + \overline{B})$
$\Rightarrow \overline{A}.B + A \Rightarrow A + \overline{A}.B \Rightarrow (A + \overline{A}).(A + B)$
$\Rightarrow A + B$

e)

$(A + C).(A.D + A.\overline{D}) + A.C + C$
$\Rightarrow (A + C).A.(D + \overline{D}) + A.C + C$
$\Rightarrow (A + C).A.A.C + C$
$\Rightarrow A.\big((A + C) + C\big) + C$
$\Rightarrow A.(A + C) + C \Rightarrow A.A + A.C + C \Rightarrow A + (A + 1).C$
$\Rightarrow A + C$

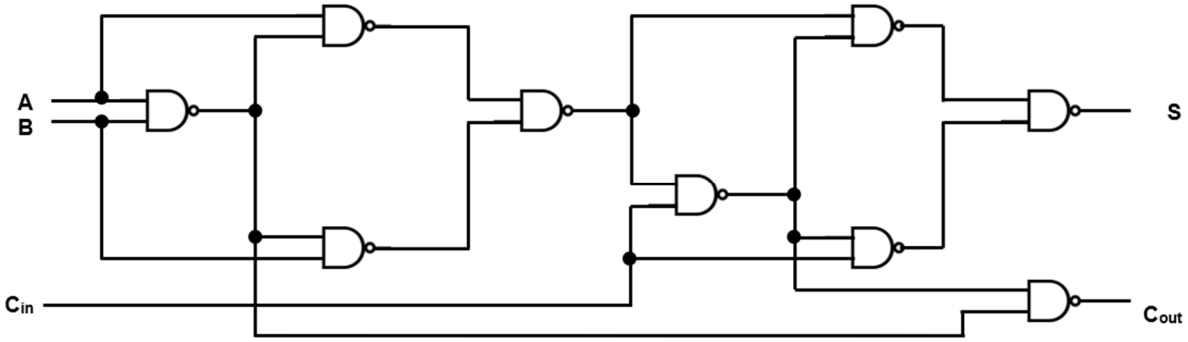
Activity 15C

1



Input		Output	
A	**B**	**S**	**C_{out}**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2



Input			Output	
A	**B**	**C_{in}**	**S**	**C_{out}**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Activity 15D

1

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

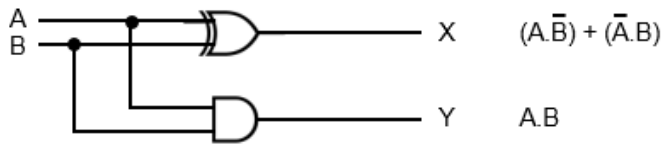
Simplified circuit is a NAND gate:

$\overline{(A . B)}$

2

A	B	X	Y
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Simplified circuit:



Activity 15E

1 a)

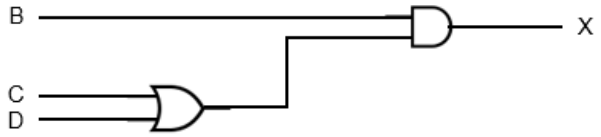
A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0

1	1	1	1	1

b) B.D + C.D = D.(C + B)

CD \ AB	00	01	11	10
00	**0**	**0**	**0**	**0**
01	**0**	**1**	**1**	**0**
11	**1**	**1**	**1**	**1**
10	**0**	**0**	**0**	**0**

c)



2 a)

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

b) B

C \ AB	00	01	11	10
0	**0**	**1**	**1**	**0**
1	**0**	**1**	**1**	**0**

3 a)

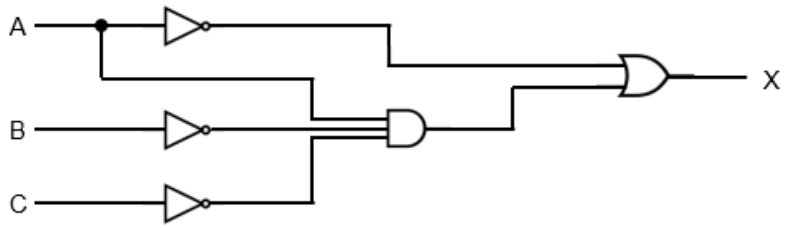
A	B	C	D	X
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1

0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

b) $A.\overline{B}.\overline{C} + \overline{A}$

CD \ AB	00	01	11	10
00	**1**	**1**	**0**	**1**
01	**1**	**1**	**0**	**1**
11	**1**	**1**	**0**	**0**
10	**1**	**1**	**0**	**0**

c)



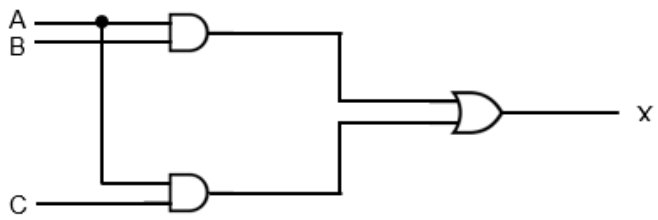
End of chapter questions

1 a) $A.B.C + A.\overline{B}.C + A.B.\overline{C}$

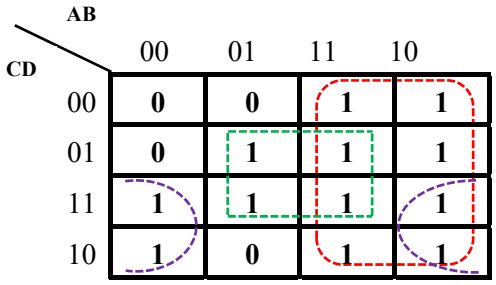
b) $A.B + A.C$

C \ AB	00	01	11	10
0	**0**	**0**	**1**	**0**
1	**0**	**0**	**1**	**1**

c)



2 a) i) ii) $A + B.D + \overline{B}.C$



b) i)

$(A + C).(A.D + A.\overline{D}) + A.C + C$
$\Rightarrow (A.C).A.(D + \overline{D}) + A.C + C$
$\Rightarrow (A + C).A + A.C + C$
$\Rightarrow A.\big((A + C) + C\big) + C$
$\Rightarrow A.(A + C) + C$
$\Rightarrow A.A + A.C + C$
$\Rightarrow A + (A + 1).C$
$\Rightarrow A + C$

ii)

$\overline{A}.(A + B) + (B + A.A).(A + \overline{B})$
$\Rightarrow \overline{A}.A + \overline{A}.B + (B + A).A + (B + A).\overline{B}$
$\Rightarrow \overline{A}.B + (B + A).A + (B + A).\overline{B}$
$\Rightarrow \overline{A}.B + B.A + A.A + B.\overline{B} + A.\overline{B}$
$\Rightarrow \overline{A}.B + B.A + A + A.\overline{B}$
$\Rightarrow \overline{A}.B + A(B + 1 + \overline{B})$
$\Rightarrow \overline{A}.B + A$
$\Rightarrow (A + \overline{A}).(A + B)$
$\Rightarrow A + B$

3 a) i)

INPUTS		OUTPUTS		
S	**R**	**Q**	$\overline{\textbf{Q}}$	**comment**
1	0	1	0	
0	0	1	0	following S = 1 change
0	1	0	1	
0	0	0	1	following R = 1 change
1	1	0	0	

ii) S = 1, R = 1, Q = 0, $\overline{Q}$ = 0 this is an invalid case since Q should be the compliment (opposite) of Q

b) i) two input values, J and K, and synchronisation (clock pulse) input

ii) uses a toggle which removes the invalid S, R states when using SR flip-flop

iii) Uses

- Several JK flip-flops can be used to produce SHIFT REGISTERS in a computer.

- A simple binary counter can be made from linking up several JK flip-flop circuits (this requires the toggle function).

4 a) SISD (single instruction single data)

- uses a single processor that can handle a single instruction which uses one data source at a time
- each task is processed in sequential order.

SIMD (single instruction multiple data)

- uses several processors which execute the same instruction set but use different data inputs
- all processes do same calculations but on different data sets simultaneously.

MISD (multiple instruction single data)

- uses several processors
- each processor uses different instructions but uses same shared data.

MIMD (multiple instruction multiple data)

- uses several processors
- each processor can accept its own instructions independently
- each processor uses data from a separate data stream (e.g. single memory which has been partitioned).

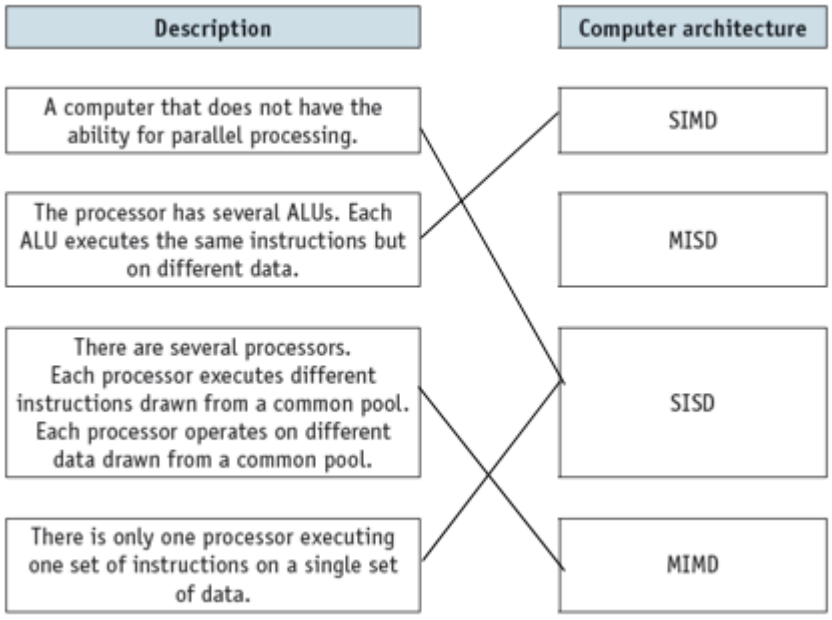
b) Features of parallel processing:

- It is a much faster way to handle large volumes of independent data.
- The data used sometimes relies on the result of a previous operation, therefore such data cannot be handled in parallel.
- Data sets require the same processing for it to work.
- It overcomes the von Neumann bottleneck and therefore greatly improves CPU performance.
- Parallel processing requires more expensive hardware.

c)

- Eight instructions need 12 clock cycles when using pipelining.
- Without pipelining, it would require 8 × 5 = 40 clock cycles to complete (each of the 8 instructions requires 5 processing stages: IF, ID, OF, IF and WB).

5 a)



b i) Massive – many processors linked together.

ii) Parallel – to perform a set of coordinated computations simultaneously.

c) Hardware – processors need to be able to communicate so that processed data can be transferred from one processor to another.

Software – suitable software which allows data to be processed by multiple processors simultaneously.

6 a) $S = \left(\overline{P} + \overline{(Q + R)}\right).R$

b)

P	Q	R	S
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

c i) ii)

R \ PQ	00	01	11	10
0	**0**	**0**	**0**	**0**
1	**1**	**1**	**0**	**0**

iii) $S = \overline{P}.R$

d)

$S = \left(\overline{P} + \overline{(Q + R)}\right).R$

$\Rightarrow S = \left(\overline{P}.(\overline{Q}.\overline{R})\right).R$

$\Rightarrow S = (\overline{P}.R) + (\overline{Q}.\overline{R}.R)$

$\Rightarrow S = \overline{P}.R + \overline{Q}.0$

$\Rightarrow S = \overline{P}.R + 0$

$\Rightarrow S = \overline{P}.R$