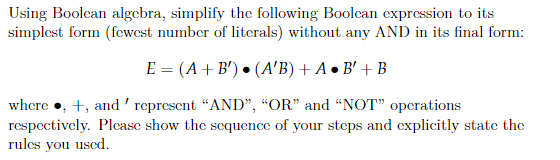
**1** **COMP40001 INTRODUCTION TO COMPUTER SYSTEMS TRA 2021-2022**

**DISCLAIMER: Take these answers with a pinch of salt, *most* of them are correct**

**1.a)**



**E = (A + B’).(A’.B) + A.B’ + B**

**E = ~~A’.B.A~~ + ~~A’.B.B’~~ + A.B’ + B Distributive Law**

**E = A.B’ + B Complement Law on AND**

**E = A + B Distributive Law [A’.B + A = B + A]**

**1.b)**

Diagram

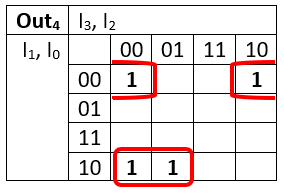
Description automatically generated

1. **Out4 used in 0, 2, 6, 8**

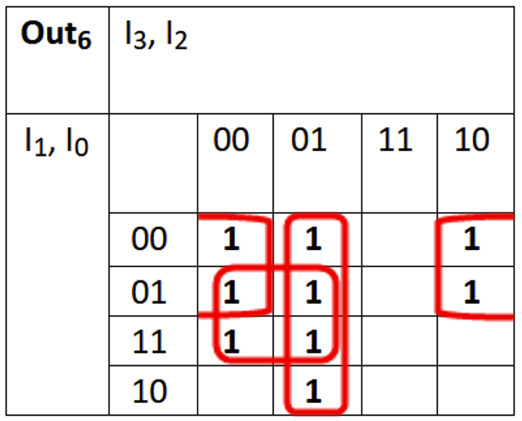
**Out6 used in 0, 1, 3, 4, 5, 6, 7, 8, 9**

**Out4 = In3’.In2’.In1’.In0’ + In3’.In2’.In1.In0’ + In3’.In2.In1.In0’ + In3.In2’.In1’.In0’**

**Out6 = In3’.In2’.In1’.In0’ + In3’.In2’.In1’.In0 + In3’.In2’.In1.In0 + In3’.In2.In1’.In0’ + In3’.In2.In1’.In0 + In3’.In2.In1.In0’ + In3’In2.In1.In0 + In3.In2’.In1’.In0’ + In3.In2’.In1’.In0**



**Out4 = In3’.In1.In0‘+ In2’.In1’.In0’**



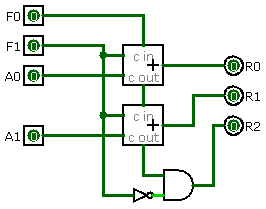
**Out6 = In3’.In0 + In2’.In1’ + In3’. In2**

**1.c)**

Text, letter

Description automatically generated

Assuming, as not specified, that the input, A, and the output, R, are unsigned. We’ve treated the case where F = 2, R = A – 1 and A = 0 as a don’t care (as there is no way to represent R = -1 in 3 bits unsigned).



**1.d)**

Text

Description automatically generated

**123.45 – 64 = 59.45**

**59.45 – 32 = 27.45**

**27.45 – 16 = 11.45**

**11.45 – 8 = 3.45**

**3.45 – 2 = 1.45**

**= 01111011.01110011001100110**

**1.45 – 1 = 0.45**

**0.45 – 0.25 = 0.2**

**0.2 – 0.125 = 0.075**

**0.075 – 0.0625 = 0.0125**

**…**

**á**

**Decimal point moves 6 to the left -> 6 = 0101**

**Sign bit = 1**

Again SharePoint is being cringe so just shift the annotations a little to the right in your head.

**= 11000010111101101110011001100110**

**C**

**2**

**F**

**6**

**E**

**6**

**6**

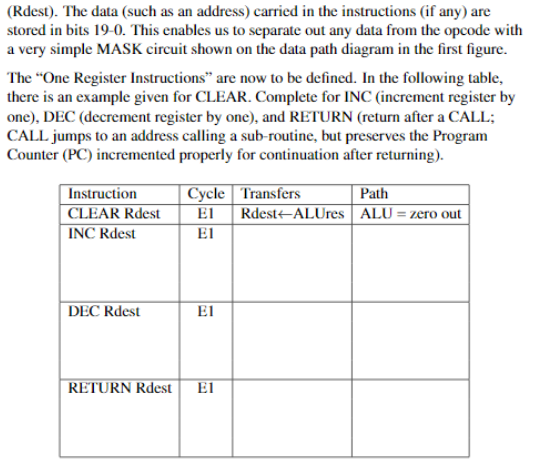
**6**

**Hex = 0xC2F6E666**

**2.a)**

Diagram

Description automatically generated



|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Cycle** | **Transfers** | **Path** |
| **CLEAR** Rdest | E1 | Rdest <- ALUres | ALU = zero out |
| **INC** Rdest | E1 | A <- Rdest |  |
|  | E2 | B <- ALUres | ALU = zero out |
|  | E3 | Rdest <- ALUres, C <- ALUcout | ALU = A + B, Cin = 1 |
| **DEC** Rdest | E1 | A <- Rdest |  |
|  | E2 | B <- ALUres | ALU = -1 |
|  | E3 | Rdest <- ALUres, C <- ALUcout | ALU = A + B, Cin = 0 |
| **RETURN** Rdest | E1 | A <- Rdest |  |
|  | E2 | PC <- SHIFTERout | SHIFTER = No action (f4=0, f3=0) |

**2.b)**

Table

Description automatically generated

Text

Description automatically generated with low confidence

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Function** | **f4** | **f3** |
| **Default** | No action | 0 | 0 |
| **ASL** | ASL | 0 | 1 |
| **ASR** | ASR | 1 | 0 |
| **X** | X | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Cycle** | **Transfers** | **Path** |
| **ASL** Rdest | E1 | A <- Rdest |  |
|  | E2 | Rdest <- SHIFTERout | SHIFTER = ASL (f4=0, f3=1) |
| **ASR** Rdest | E1 | A <- Rdest |  |
|  | E2 | Rdest <- SHIFTERout | SHIFTER = ASR (f4=1, f3=0) |

Logic shift left is the same as ASL, it serially shifts all the bits to the left while making the LSB = 0

LSR is the same as ASR but it does not preserve the MSB, it shifts everything to the right while the MSB = 0

**2.c)**

Text

Description automatically generated

Rotate left, shift left with carry

Rotate right, shift right with carry

There would be a total of 8 different shift modes (as LSL and ASL are identical), therefore there needs to be a 3-bit input into the shifter to discern the different modes.

**2.d)**



Optimise the combination logic so that each clock cycle has less delay and becomes faster.

Implement fast addition hardware to process several bits in parallel.

(Also could put:

Allow for a data path through the shifter without needing to load into A first as then all shift instructions could be done in one cycle

**2.e)**

Text

Description automatically generated

D

Q

Ck

Q’

You can make the output of a divide by 2 DQ flip flop as the input to the next divide by 2 DQ’s clock like so:

D

Q

Ck

Q’

D

Q

Ck

Q’

Therefore, it will be asynchronous as the clock of the second flip flop relies on the Q value of the first flip flop.