Embedded Systems
Professor Thomas Bräunl
Associate Lecturer Kieran Quirke-Brown

Tutorial 3 – Assembley, Circuits and Chips

1.

Registers	Initial Values	After 1.	After 2.	After 3.	After 4.
(PC)	$00	$02	$03	$05	$07
(R16)	$00	$42	$42	$42	$42
(R17)	$FF	$FF	$51	$51	$51
($004A)	$3C	$3C	$3C	$42	$42
($004B)	$1D	$1D	$1D	$1D	$51
($0050)	$42	$42	$42	$42	$42
($0051)	$B9	$B9	$B9	$B9	$B9

2.

1. LDS R16, $0400

Before	*After*
R16 =$76	R16 = *$89*
[$0400] =$89	[$0400] = *$89*

2. LDI R16, $04

Before	*After*
R16 =$76	R16 = *$04*
[$0400] =$89	[$0400] = *$89*

3. CPI R16, $76

Before	*After*
R16 =$76	R16 = *$76*
[$0400] =$89	[$0400] = *$89*
--	[CC] = ---- 0010
	(overflow flag is set to 0, negative is 0, zero is 1, carry is 0)

Note: The CPI instruction performs subtraction but does not store the result anywhere. Instead it sets the flags in the Condition Code Register.

4. LDS R16, $0400
   STS $0401, R16

Before	*After*
R16 =$76	R16 = *$89*
[$0400] =$89	[$0400] = *$89*
[$0401] =$00	[$0401] = *$89*

5. ADD R16, R17

Before	*After*
R16 =$76	R16 = $88
R17 =$12	R17 = $12

6. AND R16, R17

Before	*After*
R16 =$76	R16 = 0111 0110 AND 0001 0010 = $12
R17 =$12	R17 = $12

7. OR R16, R17

Before	*After*
R16 =$76	R16 = 0111 0110 OR 0001 0010 = $76
R17 =$12	R17 = $12

8. INC R30

Before	*After*
R30 =$79	R30 = $7A

9. DEC R30

Before	*After*
R30 =$00	R30 = $FF

10. CLR R30

Before	*After*
R30 =$FF	R30 = $00

11. SER R30

Before	*After*
R30 =$55	R30 = $FF

12. SBR R18, 1 *here: setting single bit: 0000 0001*

Before	*After*
R18 =$50	R18 = $51

13. CBR R18, 7 *here: clearing three bits: 0000 0111*

Before	*After*
R18 =$FF	R18 = $F8

14. COM R18

Before	*After*
R18 =$55	R18 = NOT 0101 0101 = $AA

15. NEG R18

Before	*After*
R18 =$55	R18 = $AA+1 = $AB

16. MOV R18, R1

Before	*After*
R18 =$55	R18 = $66
R1 =$66	R1 = $66

17. MOVW R18, R0

Before	*After*
R19 =$66	R18 = $03
R18 =$55	R18 = $02
R1 =$03	R1 = $03
R0 =$02	R0 = $02

18. LD R18,X

Before	*After*
R18 =$55	R18 = $20
X =$0450	X = $0450
[$0450] =$20	[$0450] = $20

19. LD R18,X+

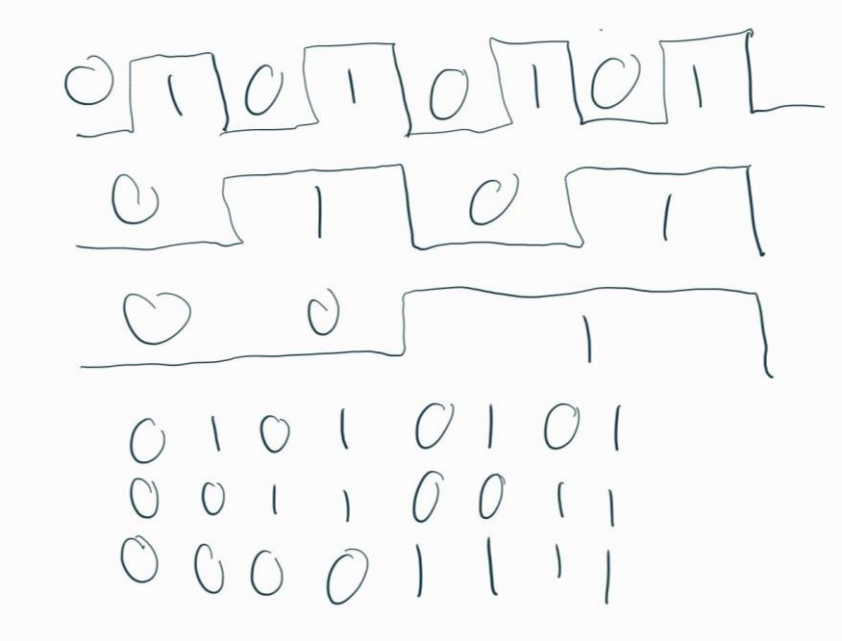
Before	*After*
R18 =$55	R18 = $20
X =$0450	X = $0451
[$0450] =$20	[$0450] = $20

20. ST -Y,R18

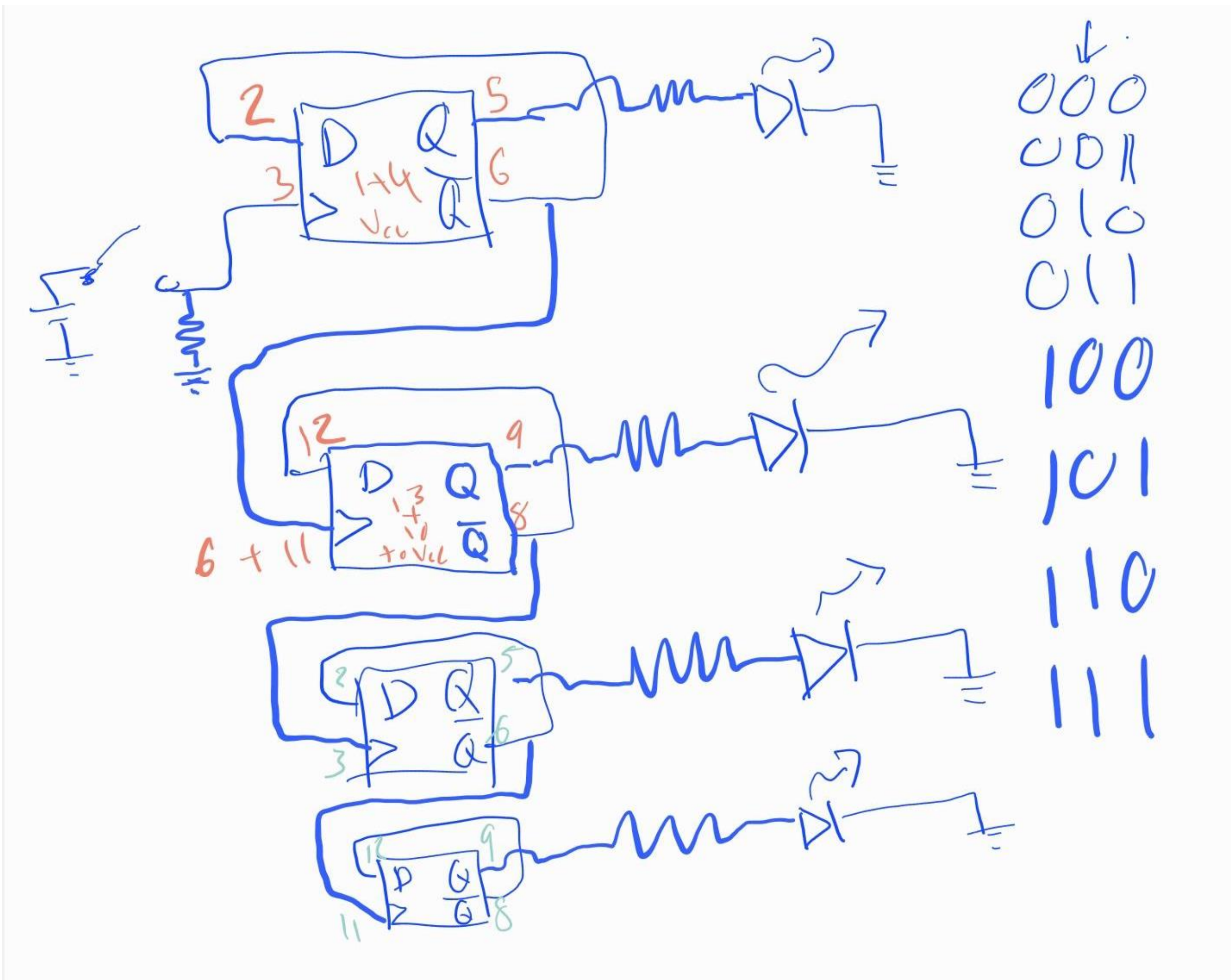
Before	*After*
R18 =$55	R18 = $55
Y =$0450	Y = $044F
[$0450] =$20	[$0450] = $20
[$044F] =$10	[$044F] = $55

3.

In order to count we need to come up with a circuit for counting. If we look at how flip flops operate we know that they only transition on a rising clock edge. We also know we can toggle the current value of a flip flop by wiring the current Q' output into D as that is the opposite to its current output.



With this we can count one bit as per the first line. Now if wanted to count higher bits we need to trigger the clock of the next flip flop when the first bit returns to 0 from 1 which means we want the opposite transition to Q. Again we can use the Q' output and pass that into the next flip flop (with it's own D wired to its own Q'). And we can see doing this over and over again will result in higher numbers to count up to. On the next page we see the circuit for this, drawn out for 4 bits. Using the data sheet for the provided chip we can find the corresponding pins for each part in the circuit and label as appropriate. In addition, we have added a pull down resistor for the input button (this value will be high, around 10KΩ). we have also added in the LEDs that also have a resistor (330Ω, so that the current through the LEDs is less than 20mA).


D
Q
2
3
5
6
1+4
Vcc
12
9
8
13
+
10
to Vcc
6 + 11
11
000
001
010
011
100
101
110
111

4.

Extra NAND chip has been included to remove debounce (on sliders this isn’t an issue but on pushbuttons it is, sliders have been use to make it easier to demonstrate)