1. **Suppose that you want to build an ALU which performs addition using 5-bit operands. For this homework you will utilize a Ripple Carry Adder as illustrated below. Assume that all digital input signals arrive simultaneously.**

**(a) [2 pts] How many OR gates will you need in order to implement the 5-bit RCA?**

Each FA contains 1 OR gate, 3 AND gates, and 2 XOR gates. So we will need 5 OR gates in order to implement the 5-bit RCA

**(b) [2 pts] How many AND gates will you need in order to implement the 5-bit RCA?**

We will need 15 AND gates in order to implement the 5-bit RCA

**(c) [2 pts] How many exclusive-OR gates will you need in order to implement the 5-bit RCA?**

We will need 10 XOR gates in order to implement the 5-bit RCA.

**(d) [3 pts] How many gate delays (abbreviated as "gds") are required in order for the RCA to compute the first sum bit ($S_0$)?**

The first sum bit S0 is available after 2 gds

**(e) [3 pts] How many gate delays (abbreviated as "gds") are required in order for the RCA to compute the second sum bit ($S_1$)?**

3 gds are required to compute S1.

**(f) [3 pts] How many gate delays are required in order for the RCA to compute the output carry bit ($C_5$)?**

C5 is available after 10 gds.

**(g) [3 pts] Imagine that you have insider knowledge regarding the value of Cin. For the application in question, you are informed that Cin will always have a value of 0. Armed with this information, how many logic gates would we be able to eliminate from the schematic? Hint: $C_1$ and $S_0$ will only depend on the value of $X_0$ and $Y_0$**

Since S0 and C1 will only depend on the value of X0 and Y0, we can eliminate the XOR gate on the bottom, and the 2 AND gates that are connected to Cin, and the OR gate because it is not necessary. Therefore, we would be able to eliminate 4 gates from the schematic.

**2. For each of the math problems below, indicate what values the AVR microcontroller will assign to each of the following flags:**

- **Carry flag**
- **oVerflow flag**
- **Negative flag**
- **Sign flag**
- **Zero flag**

**Assume that the AVR is performing 8-bit arithmetic with the built-in instructions ADD and SUB. Note that each problem is independent from the others. Feel free to check your answers using the AVR simulator, but be sure that you understand how to determine the flags.**

**(a) 21 - 32**

$21 = 0001\,0101$

$32 = 0010\,0000$

-32 can be obtained by taking 2's complement of 32

2's complement of 32 is $(0010\,0000)' + 1 = 1101\,1111 + 1 = 1110\,0000$

$21 - 32 = 21 + (-32) = 0001\,0101 + 1110\,0000 = 1111\,0101$

- C = 0 since no carry resulted
- V = 0 since there is no overflow in the 2's complement operation
- N = 1 since the result is negative
- S = 1 since N XOR V is 1
- Z = 0 since the result is not 0

**(b) 93 + 112**

$93 = 0101\,1101$

$112 = 0111\,0000$

$0101\,1101 + 0111\,0000 = 1100\,1101$

- C = 0 since no carry resulted
- V = 1 because of arithmetic overflow
- N = 1 since most significant bit is 1
- S = 0 since N XOR V is 0
- Z = 0 since the result is not 0

**(c) -3 - (-100) = -3 + 100**

$-3 = (0000\,0011)' + 1 = 1111\,1100 + 1 = 1111\,1101$

$100 = 0110\,0100$

$-3 + 100 = 1111\,1101 + 0110\,0100 = 0110\,0001$ with a carry

- C = 1 since it resulted a carry
- V = 0 since no arithmetic overflow
- N = 0 since result is positive
- S = 0 since N XOR V is 0
- Z = 0 = since result is not 0

**(d) 120 + 136**

$120 + 136 = 0111\,1000 + 1000\,1000 = 0000\,0000$ with carry

- C = 1 since carry generated
- V = 0 since no arithmetic overflow
- N = 0 since result is positive (most significant bit is 0)
- S = 0 since N OXR V is 0
- Z = 1 since result is not 1

**(e) -33 - 7**

$-33 = (0010\,0001)' + 1 = 1101\,1110 + 1 = 1101\,1111$ (taking 2's complement)

$-7 = (0000\,0111)' + 1 = 1111\,1000 + 1 = 1111\,1001$ (taking 2's complement)

$-33 + (-7) = 1101\,1111 + 1111\,1001 = 1101\,1000$ with carry

- C = 1 since carry generated
- V = 0 since no arithmetic overflow
- N = 1 since most significant bit is 1
- S = 1 since N XOR V is 1
- Z = 0 since result is not 0

**3. Problem #3 [22 pts]**

**Please read section 120 of the AVR Instruction Set Manual and review operation of the STD Z+q, Rr instruction (an indirect store with displacement).**

**(a) List and explain the sequence of microoperations required to implement this instruction on the enhanced AVR datapath (Figure 8.26 in the textbook). Note that this instruction takes two execute cycles (EX1 and EX2).**

EX1: $DMAR \leftarrow Z + q$

EX2: $M[DMAR] \leftarrow Rr$

**(b) List and explain the control signals and the Register Address Logic (RAL) output for the STD Z+q, Rr instruction. Clearly explain your reasoning. Control signals for the Fetch cycle are given below.**

***Control signals:***

During EX1, ZH and ZL are simultaneously fetched from the Register File and then fed to the Address Adder by selecting input-1 of the MUXG. The content of Z is then passed to the output by setting adder_f to 11. This output then becomes the input of DMAR by setting MUXH to 1. At the same time, DM_w, RF_wA, and RF_wB are set to 0 to keep the Data Memory and Register File from being updated. To prevent PC and IR from being overwritten, PC_en and IR_en are also set to 0.

During EX2, the effective address is latched on to DMAR by setting MUXE to 1 so that it can be used to access the Data Memory. Rr is read from register file and then fed to the Data Memory as input by setting MUXD to 1, DM_w to 1 and DM_r to 0. RF_wA, RF_wB, IR_en, PC_en are set to 0. The other signals can be “don’t care”

***RAL:***

During EX1, YH and YL are mapped to rA and rB, respectively. During EX2, Rd from the instruction format serves as the source register identifier, and is mapped to rB.

| Control Signals | IF | STD Z+q, Rr | |
|---|---|---|---|
| | | EX1 | EX2 |
| MJ | 0 | X | X |
| MK | 0 | X | X |
| ML | 0 | X | X |
| IR_en | 1 | 0 | X |
| PC_en | 1 | 0 | 0 |
| PCh_en | 0 | 0 | 0 |
| PCl_en | 0 | 0 | 0 |
| NPC_en | 1 | X | X |
| SP_en | 0 | 0 | 0 |
| DEMUX | x | X | X |
| MA | x | X | X |
| MB | x | X | X |
| ALU_f | xxxx | xxxx | xxxx |
| MC | xx | xx | xx |
| RF_wA | 0 | 0 | 0 |
| RF_wB | 0 | 0 | 0 |
| MD | x | X | 1 |
| ME | x | X | 1 |
| DM_r | x | X | 0 |
| DM_w | 0 | 0 | 1 |
| MF | x | X | X |
| MG | x | 1 | X |
| Adder_f | xx | 11 | xx |
| Inc_Dec | x | X | X |
| MH | x | 1 | X |
| MI | x | X | X |

| RAL Output | STD Z+q, Rr | |
|---|---|---|
| | EX1 | EX2 |
| wA | X | X |
| wB | X | X |
| rA | ZH | X |
| rB | ZL | Rd |