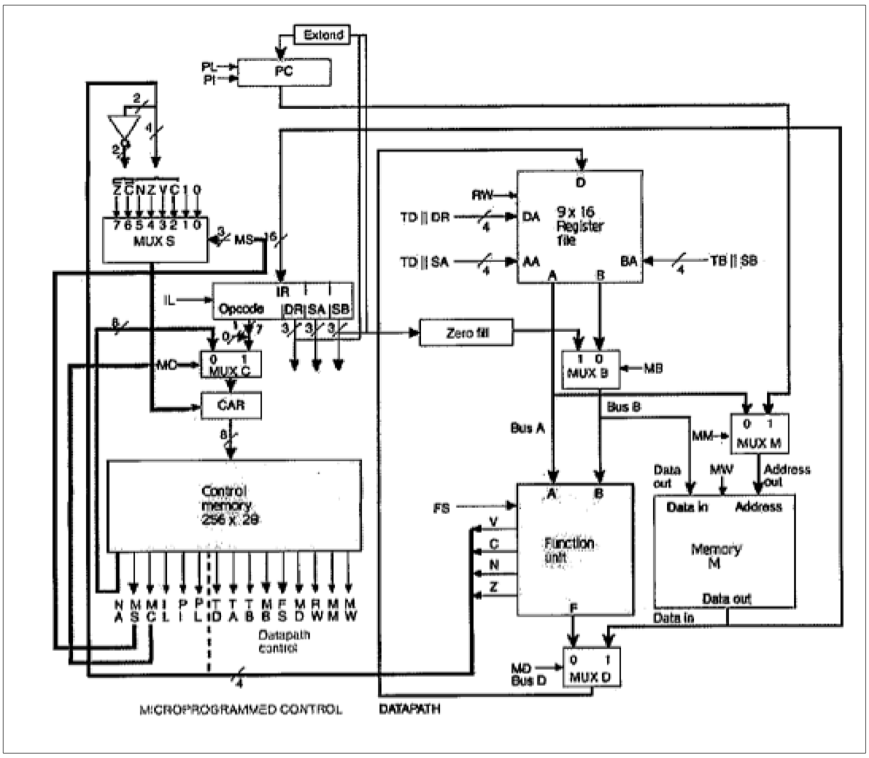
Computer Architecture – 2017 Solutions

Question 1

(1)

a) Explain in detail the operations that take place when the following multiple cycle microprogrammed instruction set processor executes machine instructions. Your explanation must include operations in the processor's control e.g how does one instruction in the IR register execute several control words in the Control Memory.



Setup

When the system is first turned on the Program Counter (PC) and the Control Access Register (CAR) are **RESET**. This is to ensure all addresses and values within these devices are reset to the desired start off values.

The Program Counter (PC) specifies the address of the next microprogram stored within the Memory Unit (M), while the Control Access Register (CAR) specifies the address of the next set of machine code instructions within the Control Memory. The PI signal increments the value of the PC, and the PL signal uses DR || SB (concatenated) as the next address.

When the PC is **RESET** it is loaded with the address of the first microprogram within the Memory Unit (M). Memory contains the addresses of instructions that are stored within the Control Memory. For example, instruction 0x002E from the Memory Unit (M) may correspond to the Micro Code for an ADI instruction located at a given address within the Control Memory.

Execution

The microprogram (instruction) address is loaded from the Memory Unit (M) into the Instruction Register (IR), where it is decoded and split into a 7-bit Opcode that corresponds to the address of the instruction to be accessed within the Control Memory. It is also split into 3 x 3-bit vectors that can either be used for register selection or as values (either immediate or PC offset for branching). They decide the Destination Register (DA).

Opcode	DA	SA	SB
7 bit	3 bit	3 bit	3 bit

For example

`ADD r0, r1, r2 =`

0000011	000	001	010

`= 0000011000001010    = 0x060A`

The Opcode is then loaded into MUXC, and, as the control memory is in the Instruction Fetch mode, it is loaded through MUXC and into the Control Access Register (CAR). Here the Opcode is used to determine the next address the Control Memory uses to access the next instruction.

The Control Memory accesses the instruction to be accessed (as dictated by the CAR) based on the instruction address passed in. An instruction could take more than one clock cycle (e.g LOAD).

The Control Memory controls the input signals to all of the multiplexers, Function Unit, Register File, PC, IR, CAR and Memory.

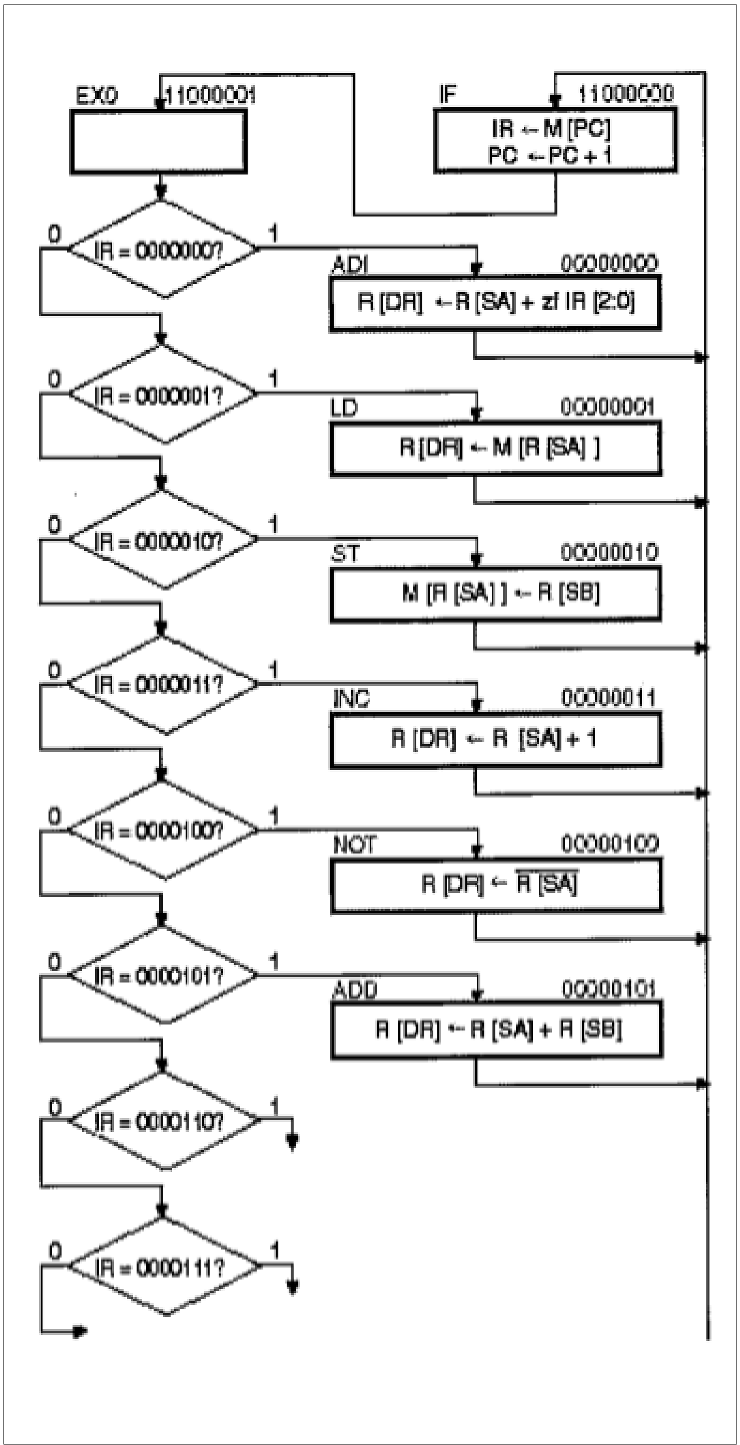
Values coming from either the output of the Function Unit or Memory (decided by MUXD) can be loaded into registers within the Register File if RW (Read=0/Write=1) is set to high. The Register File contains an array of registers which hold data. A temporary register (R8) is also here for multi-cycle instructions so that the data the user may be storing is not disrupted. The Destinartion Register is selected by DR coming from IR, or TD (for temp register) coming from Control Memory.

The Function Unit takes in values from the Register File (selected by SA, SB, TA or TB) or immediate values (passed from IR through MUXB). It then performs arithmetic, logic or shifting operations on these values. It contains an Arithmetic Logic Unit (ALU) and a Shifter. The status bits from the Function Unit are passed through to MUXS and are used for conditional instructions within the Microprogrammed Control.

Values from the Register File can also be written and stored into the Memory Unit (M) if MW (Memory Write) is set to high.

The processor is pipelined so that a number of operations can be happening simultaneously. The system could simultaneously be fetching the next instruction, executing a previous instruction and writing the results to the Register File.

a) Provide Micro Code for the following instructions:



1. *IF – Instruction Fetch :*

   Fetch the Instruction at M[PC] and load the value into IR. Increment PC after.

2. EX0 – Execute:

   Execute the instruction fetched.

3. *ADI – Add Immediate Operand:*

   Add the value in R[SA] with the immediate operand in IR[2:0]. Store result in R[DR]

4. *LD – Load*

   Load, from memory to R[DR] the contents within Memory Unit M at M[R[SA]]

5. *ST – Store*

   Store, to memory at M[R[SA]] the value within the register R[SB]

6. *INC – Increment*

   Increment the value of R[SA] by 1 storing the result in R[DR]

7. *NOT – Not*

   Invert R[SA], storing result in R[DR]

8. *ADD – Add A + B*

   Add the contents of R[SA] with the contents of R[SB] and store the results in R[DR]

Microinstruction Format:

27 26 25 24 23 22 21 20	19 18 17	16	15	14	13	12	11	10	9	8 7 6 5 4	3	2	1	0
NA	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW

1. IF – Instruction Fetch – [1100 0000]

Fetch the Instruction at M[PC] and load the value into IR. Increment PC after.

Address in Control Memory – [1100 0000]

Instruction Description -

- Set IL - Instruction Load into the IR
- Set MM - Passes PC value through to Memory Unit M
- Set PI – Increments Program Counter (PC)
- NA = EX0 – Execute the Instruction Loaded – [1100 0001]

NA	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
1100 0001	000	0	1	1	0	0	0	0	0	00000	0	0	1	0

2. EX0 – Execute the Instruction Loaded – [1100 0001]

Fetch the Instruction at M[PC] and load the value into IR. Increment PC after.

Address in Control Memory – [1100 0000]

Instruction Description -

- Set MC – Allow IR Opcode to pass through to CAR
- Set MS – Set MS select to 1, pass 1 through to CAR
- NA = ADI – Add Immediate Operand – [0000 0000]

NA	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
0000 0000	001	1	0	0	0	0	0	0	0	00000	0	0	0	0

3. ADI – Add Immediate Operand – [0000 0000]

Add the value in R[SA] with the immediate operand in IR[2:0].

Address in Control Memory – [0000 0000]

Instruction Description -

- Set MB – Select Immediate Operand (IR[2:0]) as B
- FS = 00010 – FS = A+B
- Set MD to 0 – Select Function Unit out as Bus D
- Set RW to 1 – Set Read/Write to Write
- NA = DONT CARE

NA	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
0000 0000	000	0	0	0	0	0	0	0	1	00000	0	1	0	0

4. *LD – Load – [0000 0001]*

Load, from memory to R[DR] the contents within Memory Unit M at M[R[SA]]

Address in Control Memory – [0000 0001]

Instruction Description -

- Set RW to 0 – Read
- Set MM to 0 – Take Bus A (R[SA]) and pass to Memory M
- Set MW to 0 – Read from Memory
- Set MD to 1 – Select Memory out as Bus D
- Set RW to 1 – Set Read/Write to Write
- NA = DONT CARE

NA	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
0000 0000	000	0	0	0	0	0	0	0	0	00000	1	1	0	0

5. *ST – Store – [0000 0010]*

Store, to memory at M[R[SA]] the value within the register R[SB]

Address in Control Memory – [0000 0010]

Instruction Description -

- Set MB to 0 – Select R[SB] as Bus B
- Set MM to 0 – Select R[SA] as Address within Memory
- Set MW to 1 – Memory Write
- Set RW to 1 – Set Read/Write to Write
- NA = DONT CARE

NA	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
0000 0000	000	0	0	0	0	0	0	0	0	00000	0	1	0	1

6. *INC – Increment – [0000 0011]*

Increment the value of R[SA] by 1 storing the result in R[DR]

Address in Control Memory – [0000 0011]

Instruction Description -

- Set FS = 00001 (F = A + 1)
- Set MD to 0 – Select Function Unit out as Bus D
- Set RW to 1 – Set Read/Write to Write
- NA = DONT CARE

NA	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
0000 0000	000	0	0	0	0	0	0	0	0	00001	0	1	0	0

7. *NOT – Not – [0000 0100]*

Invert R[SA], storing result in R[DR]

Address in Control Memory – [0000 0100]

Instruction Description -

- Set FS = 01110 (F = A’)
- Set MD to 0 – Select Function Unit Out as Bus D
- Set RW to 1 – Set Read/Write to Write
- NA = DONT CARE

NA	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
0000 1000	000	0	0	0	0	0	0	0	0	01110	0	1	0	0

8. *ADD – Add A + B – [0000 0101]*

Add the contents of R[SA] with the contents of R[SB] and store the results in R[DR]

Address in Control Memory – [0000 0101]

Instruction Description -

- Set FS = 00010 (F = A + B)
- Set MD to 0 – Select Function Unit Out as Bus D
- NA = Store Bus D to R[DR]- [0000 1000]

NA	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
0000 1000	000	0	0	0	0	0	0	0	0	00010	0	0	0	0

Question 2

(2)

a) Provide a detailed schematic for a Function Unit that implements the following micro operations:

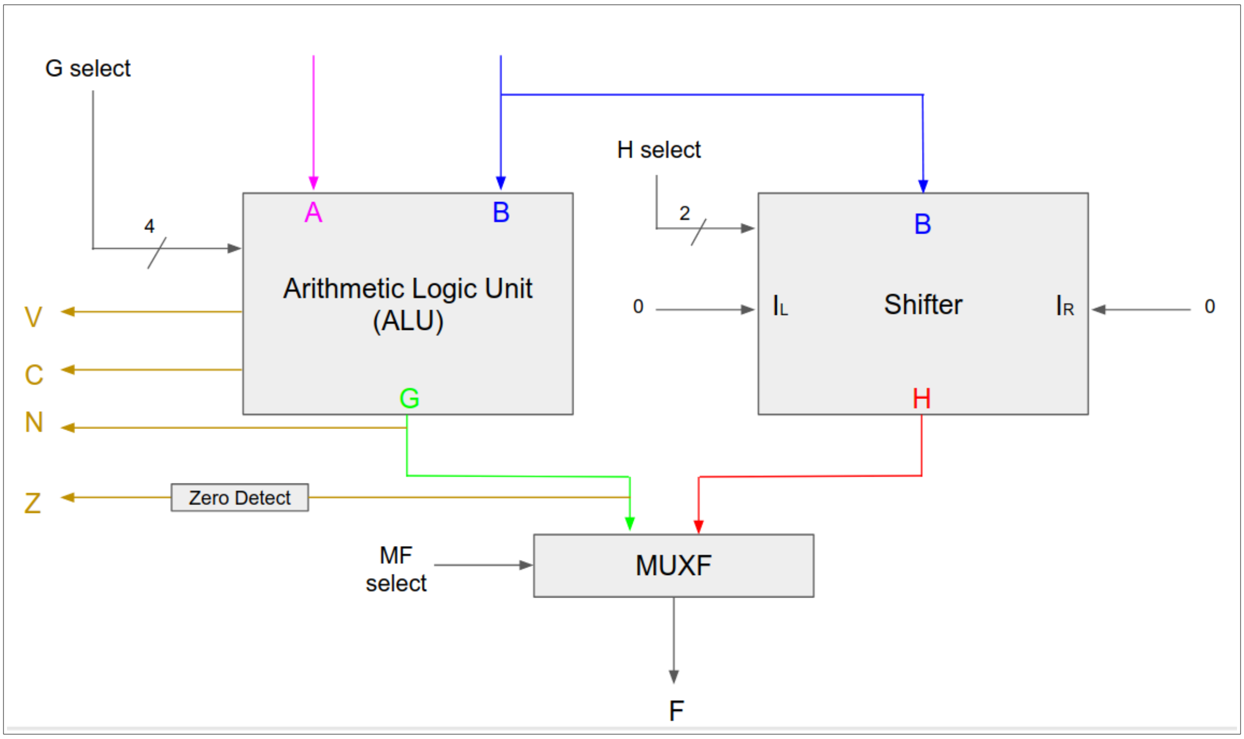
Table 1: FS code definition

FS	Micro-operation
00000	$F = A$
00001	$F = A + 1$
00010	$F = A + B$
00011	$F = A + B + 1$
00100	$F = A + \bar{B}$
00101	$F = A + \bar{B} + 1$
00110	$F = A - 1$
00111	$F = A$
01000	$F = A \wedge B$
01010	$F = A \vee B$
01100	$F = A \oplus B$
01110	$F = \bar{A}$
10000	$F = B$
10100	$F = srB$
11000	$F = slB$

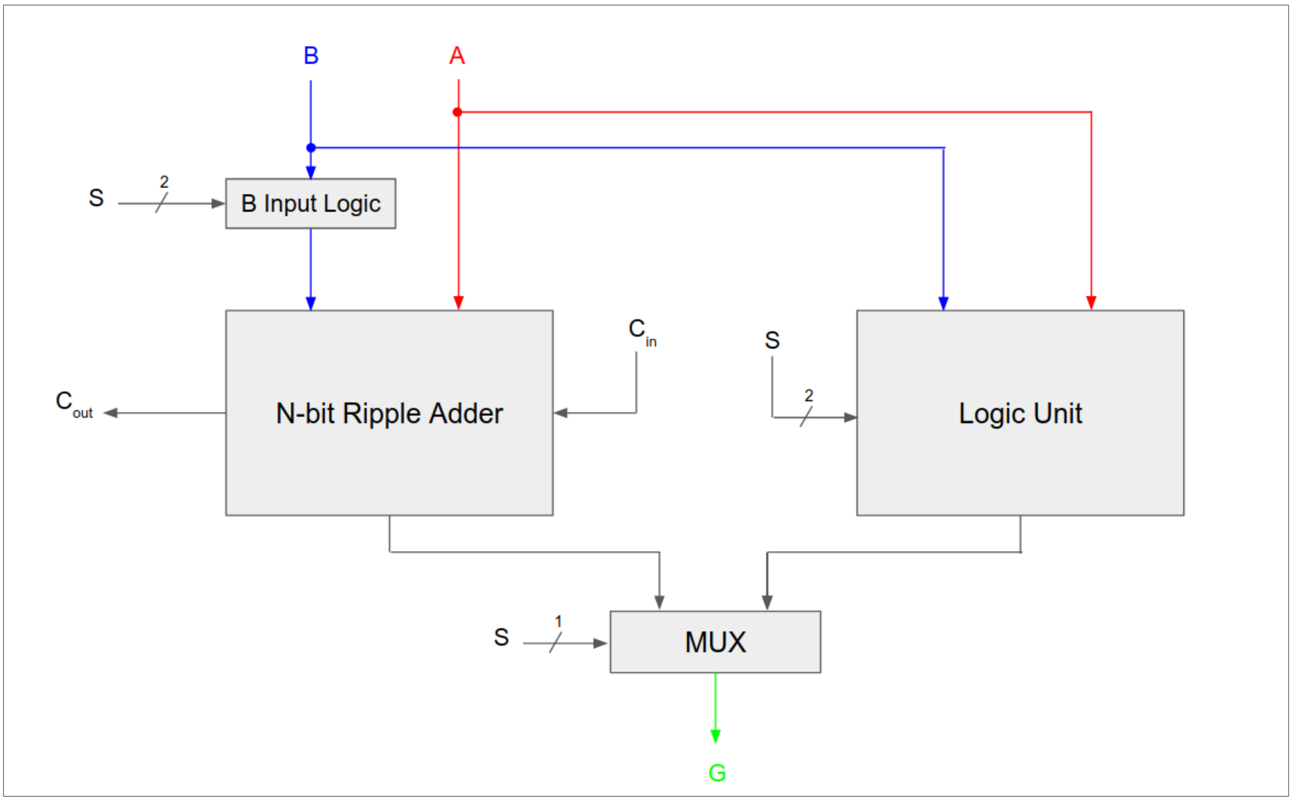
A Function Unit consists of the following components:

- 16bit Arithmetic Logic Unit (ALU):
  - N-Bit Ripple Adder
    - N x Full Adders
  - Logic Unit
  - B Input Logic
- 16bit Shifter:
- 2 to 1 line multiplexer:

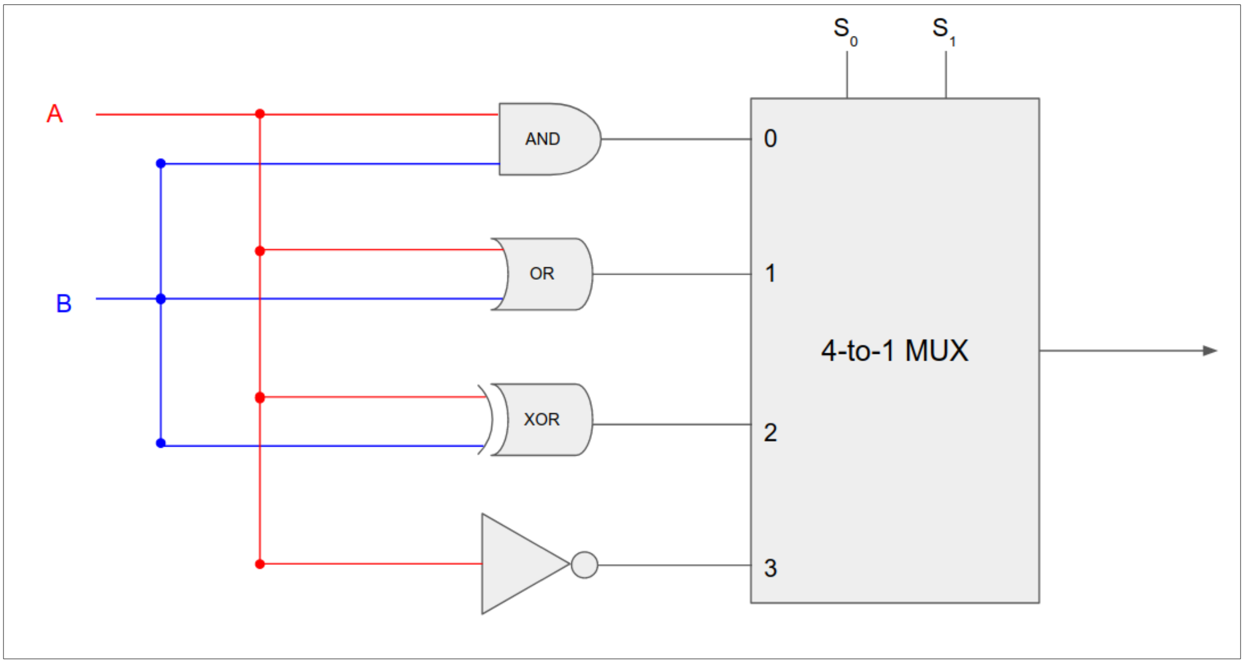
1. *Function Unit Overview*



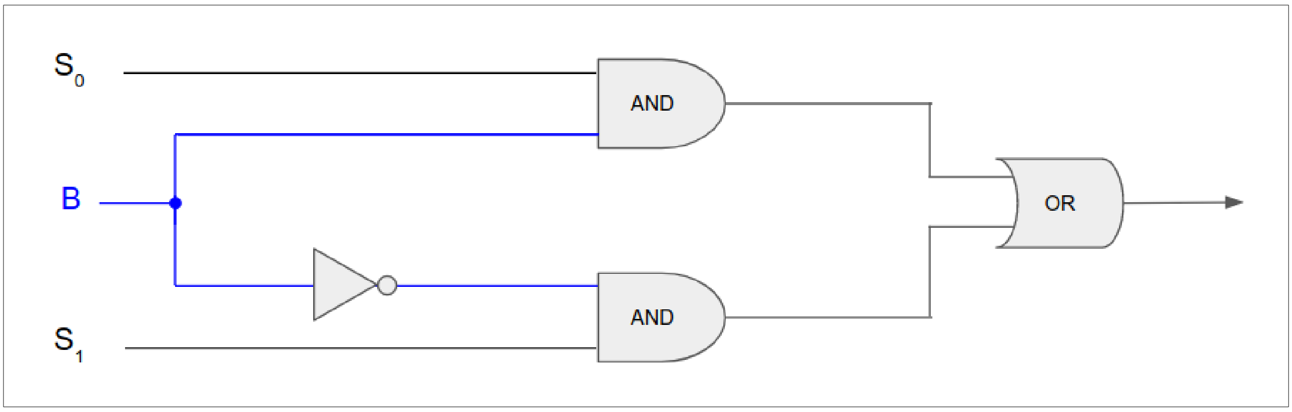
2. 16-Bit Arithmetic Logic Unit (ALU)



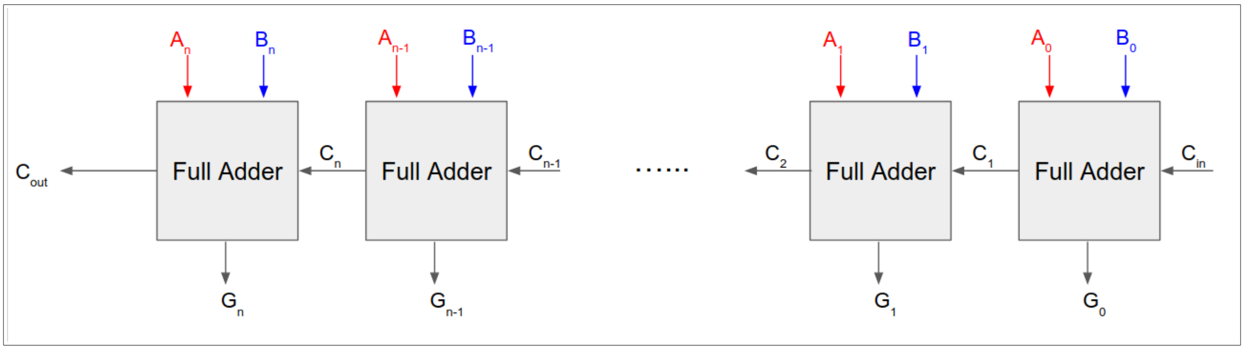
3. Logic Unit



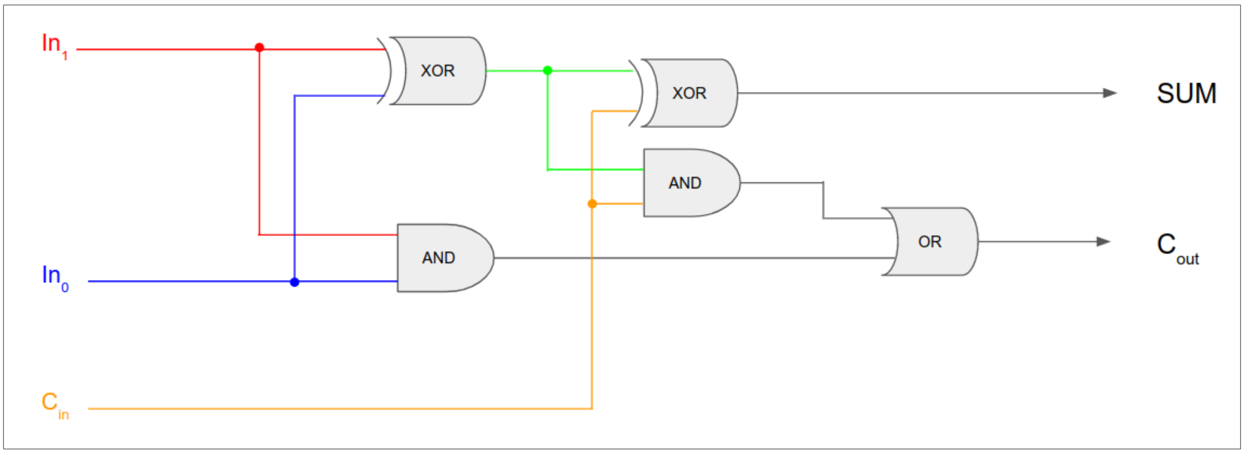
4. B Input Logic (1-bit Slice)



5. Ripple Adder



6. Ripple Adder



7. 16-Bit Shifter Schematic