b) What modifications to the ASM chart are necessary to implement the Binary Multiplier as a microcoded control solution?

[5 marks]

b) Normally we store the control words that hold the next-state information into control memory and then use control input and status signal from control memory to select the address of the next state.

Since we don't store control words, we can't depend on control input to find our next instruction. Instead, additional states are supplied.

An initiate state and a load state would be added, meaning we have 5 states. A 3 bit control address register is introduced to manage this.

A multiplexer is used to access the control input, where the result determines which of the two store addresses is used.

(After the description of the operations when an instruction executes that comes up as a part A every year)

b) Expand on your discussion from Question 2.a) by explaining how the following instruction is executed, Immediate Instruction

[5 marks]

b)

1. Fetch next instruction
    a. Increment PC and load in from Memory M
    b. Write instruction to IR (Instruction Register)
    c. Write opcode from IR to CAR (Control Access Register)
2. Access control word of ADI in Control Memory
3. Set the control signals that go from Control Memory to MUX's and other bits of the microprogram (IR, FUnction Unit etc)
    a. MB = 1
    b. RW =1
    c. FS=00010
    d. MD=0
    e. MC=0
    f. MS-
    g. NA=Next control word address
    h. PI=0
    i. PL=0
    j. IL=0

k. MM=X

b) What do the following Boolean Expressions implement? Please provide a detailed discussion.

$C_{i+1} = g_i + p_iC_i$

$C_1 = x_0y_0 + C_0(x_0 + y_0)$

$\quad = g_0 + C_0p_0$

$C_2 = x_1y_1 + C_1(x_1 + y_1)$

$\quad = x_1y_1 + [x_0y_0 + C_0(x_0 + y_0)](x_1 + y_1)$

$\quad = g_1 + p_1g_0 + p_0p_1C_0$

$C_3 = g_2 + p_2g_1 + p_1p_2g_0 + p_0p_1p_2C_0$

$C_4 = g_3 + p_3g_2 + p_2p_3g_1 + p_1p_2p_3g_0 + p_0p_1p_2p_3C_0$

[5 marks]

This equation denotes a carry look-ahead adder.

The first line $C_{i+1} = g_i + p_iC_i$ is the carry out for each stage of the addition.

In a *ripple adder* the carry for each stage is calculated by checking the carry of the previous stage. This leads to a propagation delay as the adder takes some time to figure out the carry, and so the next adder must wait for it to figure it out. This = propagation delay.

In a *carry look-ahead adder* a the full carry is calculated separately to the adders. Which means the next stage does not have to wait for a carry to be calculated. It’s just added on at the end. At each stage in carry look ahead the equation is fully expanded which leads to more redundant gates but also limit propagation delay.

In essence carry look ahead calculates if they’ll be a carry before the addition occurs, ripple adders have to wait for the previous carry to be generated before C_{out} becomes the next C_{in}

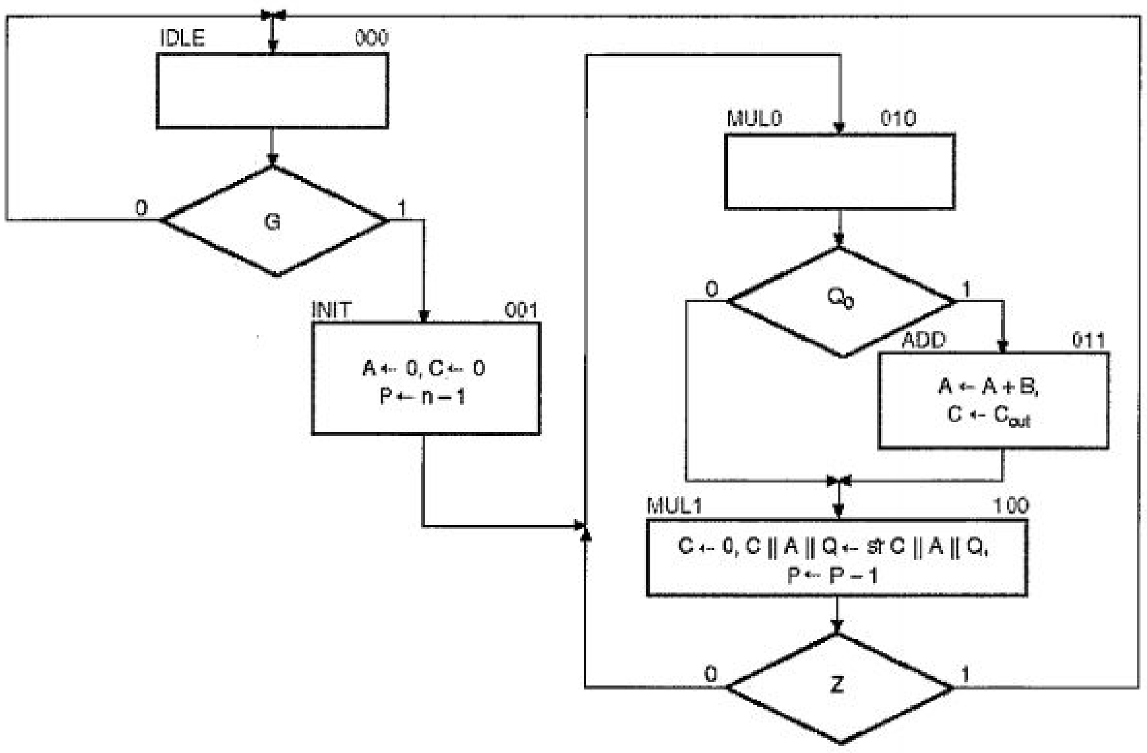
b) Discuss the register transfer operations that can be performed with this Register-file.

[5 marks]

The register file can take in external data from input lines, one register at a time.
It can also copy the content of one register to another.
The data_src input decides whether external source or register source data will be loaded into the destination register (DR).

On each clock cycle (if instructed) one of the registers will be updated in the register file will be updated.

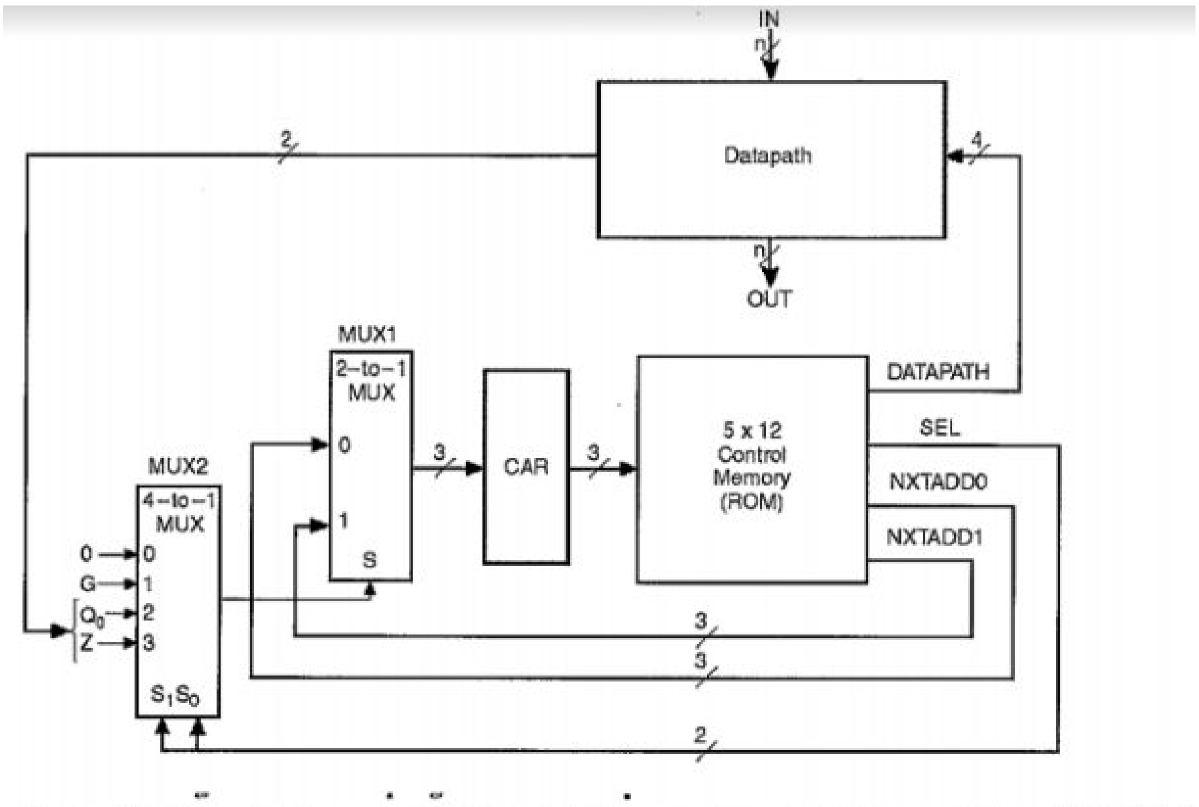
(Binary Multiplier - 2014 Q A)



a) The above ASM Chart is suitable for a microcoded control implementation. What would you need to change in order to make the ASM suitable for a hardwired solution? Why is this modification necessary?

[5 marks]

a) Change the state boxes (INIT and ADD) to D flip flops (or conditional output boxes) that can hold a state value indefinitely. They control the state that the Binary Multiplier is in.
b) The decision boxes (diamonds) would be change to demultiplexers. Decisions would be made with decision signals and output would be the state of the state box to be store in the D flip flops.
c) Junctions (where wires meet) would become OR gates. If there is a result to be given they would allow either to be used (or both!)
d) Add output signals.

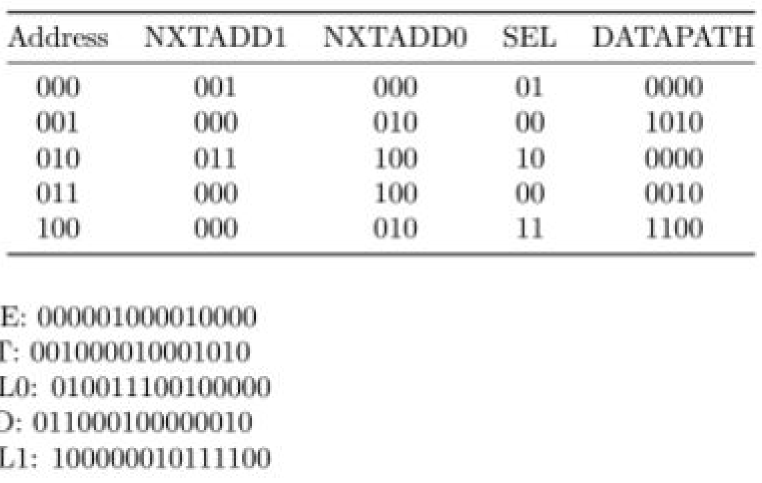


Control Signal	Register Transfers	States in Which Signal is Active	Micro-instruction Bit Position	Symbolic Notation
Initialize	$A \leftarrow 0, P \leftarrow n-1$	INIT	0	IT
Load	$A \leftarrow A+B, C \leftarrow C_{out}$	ADD	1	LD
Clear_C	$C \leftarrow 0$	INIT, MUL1	2	CC
Shift_dec	$C\|A\|Q \leftarrow \text{sr } C\|A\|Q, P \leftarrow P-1$	MUL1	3	SD

b) Please provide microcode for the above Control Memory (5x12) that implements the above ASM Chart. Discuss how the control hardware executes your microcode.

[15 marks]

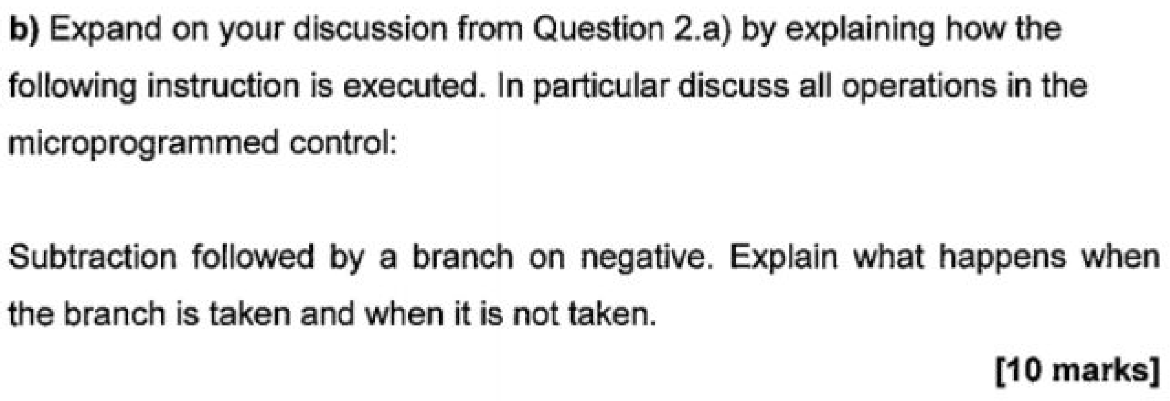
State	Address		NXTADDR1	NXTADDR0	SEL	SD	CC	LD	IT
IDLE	000		001	000	01	0	0	0	0
INIT	001		000	010	00	0	1	0	1
MUL0	010		011	100	10	0	0	0	0
ADD	011		000	100	00	0	0	1	0
MUL1	100		000	010	11	1	1	0	0



Address	NXTADD1	NXTADD0	SEL	DATAPATH
000	001	000	01	0000
001	000	010	00	1010
010	011	100	10	0000
011	000	100	00	0010
100	000	010	11	1100

- IDLE: 000001000010000
- INIT: 001000010001010
- MUL0: 010011100100000
- ADD: 011000100000010
- MUL1: 100000010111100

(The question after the big diagram & spiel that comes up every year 2014)



b) Expand on your discussion from Question 2.a) by explaining how the following instruction is executed. In particular discuss all operations in the microprogrammed control:

Subtraction followed by a branch on negative. Explain what happens when the branch is taken and when it is not taken.

[10 marks]

1. The instruction is fetched
    a. The PC is incremented
    b. The instruction is passed to the IR (Instruction Register)
    c. Opcode passed from IR to CAR (Control Access Reg) and 3 registers for the register file are selected
2. The subtraction microprogram begins once the address is retrieved from Control memory
    a. The values of A & B are given to the functional unit from the Register File
    b. Control Memory give the functional unit the control signals for subtraction and the correct arithmetic circuit is chosen using a MUX in the ALU.
    c. The result is saved into the DR(Destination register)
3. The next instruction (BNE Branch Negative) is loaded in and the N flag is loaded into the CAR
    a. If N=0 then increment the CAR and get the next instruction as regular
    b. If N=1 then go to the memory address that stores the microprogram for unconditional branching
        i. Thee PL flag would be set and the memory address would be determined by the concatenation of DR and SB
        ii. The fetch next instruction microprogram would begin