EE 224

IITB CPU

PEN-PAPER DESIGN

Group

Advay Bapat 23B1225

Jigar Mehta 23B1273

Saniya Khinvasara 23B1268

Sukeshi Patni 23B1299

Page No.
Date

Arithmetic Logic Instructions.

I] ADD / SUBTRACT / LOGICAL AND / LOGICAL OR / MUL / IMP

S0 -

PC ⟶ Mem-Add.	Mem-Read.
Mem-D ⟶ IR	ADD.
PC ⟶ ALU-A.	PC-en.
+1 ⟶ ALU-B.	IR-en.
ALUC ⟶ PC.	

S1 -

$IR_{11-9} \longrightarrow RF\_A_1$	
$IR_{8-6} \longrightarrow RF\_A_2$	T_1-en.
$RF\_A_1 \longrightarrow T_1$	T_2-en.
$RF\_D_2 \longrightarrow T_2$.	

S2 -

$T_1 \longrightarrow$ ALU-A.	ADD / SUB / MUL / AND / OR /
$T_2 \longrightarrow$ ALU-B	IMP.
ALUC ⟶ T_1	

S3 -

$T_1 \longrightarrow RF\_D_3$	RF_en.
$IR_{5-3} \longrightarrow RF\_A_3$	

S0 ⟶ S1 ⟶ S2 ⟶ S3

Page No.
Date

II] Add Immediate (ADI)

S0 -

PC ⟶ Mem-Add	Mem_Read
Mem-D ⟶ IR.	PC-en.
PC ⟶ ALU-A.	ADD.
+2 ⟶ ALU-B	IR-en.
ALU-C ⟶ PC	

S1 -

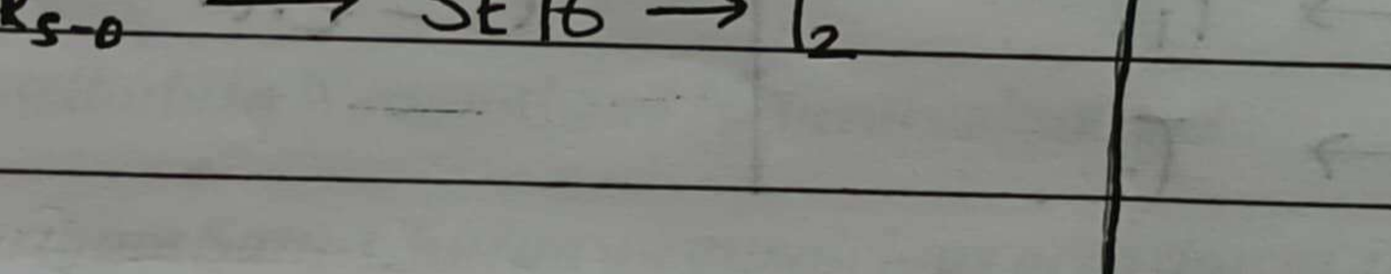
IR_{11-9} ⟶ RF_A_1	T_1-en.
RF-D_1 ⟶ T_1	T_2-en.
IR_{5-0} ⟶ SE16 ⟶ T_2	

S2 -

T_1 ⟶ ALU-A.	ADD
T_2 ⟶ ALU-B	T_1-en
ALU-C ⟶ T_1	

S3-

T_1 ⟶ RF-D_3	RF-en
IR_{8-6} ⟶ RF-A_3	



LHI - Load High Immediate - 1000

S_0

PC ⟶ Mem_Add	PC_en
PC ⟶ ALU_A	IR_en
2 ⟶ ALU_B	Mem_Read
ALU_C ⟶ PC	ALU_Add
Mem_Out ⟶ IR	

S_1

IR_{11-9} ⟶ RF_A3	RF_en
IR_{8-0} ⟶ SE16 ⟶ 8S ⟶ RF_D3	

LLI - Load Lower Immediate - 1001

S_0 ⟶ Same as before

S_1

IR_{11-9} → RF_A3	RF_en
IR_{8-0} ⟶ RF_D3	

LW - Load - 1010

S_0 ⟶ Same as before

S_1 →

IR_{8-6} ⟶ REA2	T_2_en
RED2 ⟶ T2	

S_2 →

$T_2 \longrightarrow$ ALU-A

$IR_{5-0} \longrightarrow$ SE16 $\longrightarrow$ 15 $\longrightarrow$ ALU-B

ALU-C $\longrightarrow$ T_2

| T_2-en
| ALU-Add

S_3 →

$T_2 \longrightarrow$ Mem_Add

Mem_Out $\longrightarrow$ T_2

| T_2-en
| Mem_Read

S_4 →

$IR_{11-9} \longrightarrow$ RF_A3

$T_2 \longrightarrow$ RF_D3

| RF_en

SW - Store - 1011

$S_0 \rightarrow$ Same as before

S_1

$IR_{11-9} \longrightarrow$ RFA1

$IR_{8-6} \longrightarrow$ RFA2

RF_D2 $\longrightarrow$ T2

RF_D1 $\longrightarrow$ T1

| T_2-en
| T_1-en

S_2

$T_2 \longrightarrow$ ALU-A

$IR_{5-0} \longrightarrow$ SE16 $\longrightarrow$ 15 $\longrightarrow$ ALU-B

ALU-C $\longrightarrow$ T_2

| T_2-en
| ALU-Add

S_3

$T_2 \longrightarrow$ Mem_Add.

$T_1 \longrightarrow$ Mem_In

| Mem_en

JUMP AND LINK. (JAL)

S0 —	
PC → Mem-Add Mem-D → IR PC → ALU-A. +2 → ALU-B ALU-C → PC	Mem-Read IR-en. PC-en. ADD

S1-	
PC → ALU-A. +2 → ALU-B ALU-C → T_2	SUB T_2-en

S2 -	
T_2 → RF-D_3 IR_{11-9} → RF-A_3	RF-en.

S3 -

T_2 → ALU-A IR_{8-0} → SE16 →	shift left	→ ALU-B	ADD PC-en.

ALU-C → PC

$S_0 \longrightarrow S_1 \longrightarrow S_2 \longrightarrow S_3$

Page No.
Date

JLR — Jump And Link To Register

S0 —

PC → Mem_Add	Mem_Read
Mem_D → IR	IR_en
PC → ALU_A	PC_en.
+2 → ALU_B	ADD
ALU_C → PC	

S1 —

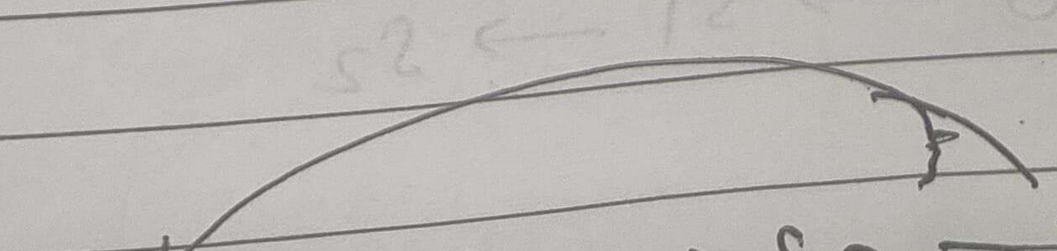
PC → ALU_A.	SUB
+2 → ALU_B	T_2_en.
ALU_C → T_2	

S2 —

T_2 → RF_D_3	RF_en.
IR_{11-9} → RF_A3	

S3 —

IR_{8-6} → RF_A_1	PC_en
RF_D_1 → PC	



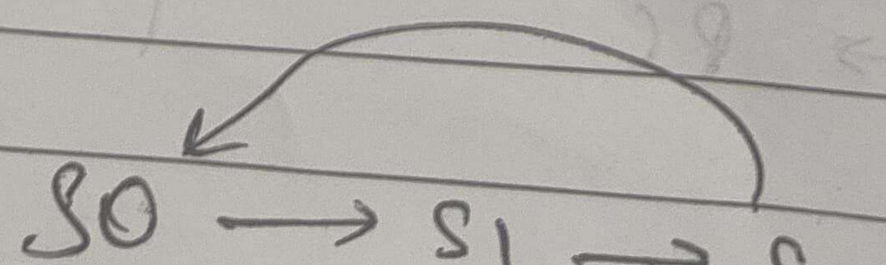
Page No.
Date

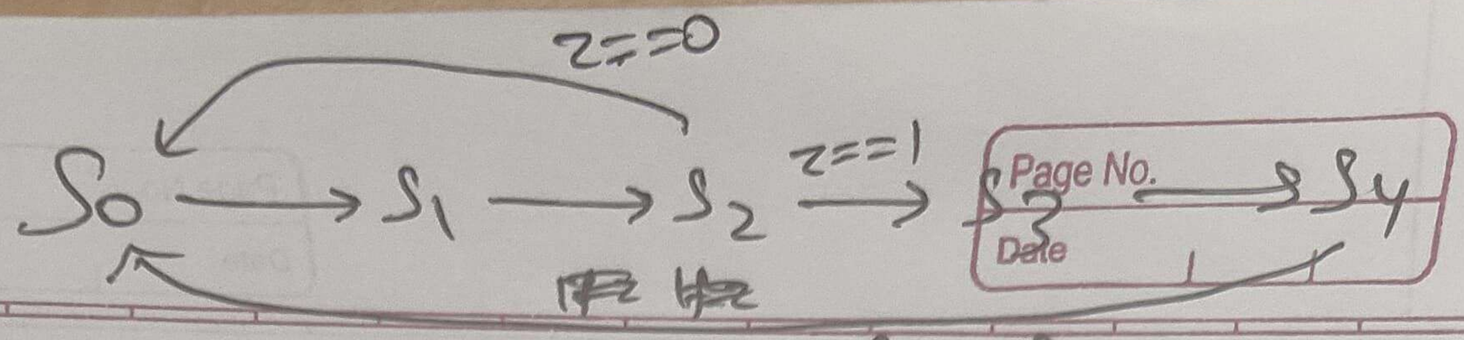
JUMP UNCONDITIONALLY (J)

S0 —	
PC → Mem_Add	Mem_Read
Mem_D → IR	IR_en
PC → ALU_A	ADD
+2 → ALU_B	PC_en.
ALU_C → PC	

S1 —	
PC → ALU_A.	SUB
+2 → ALU_B	T2_en.
ALU_C → T2	

S2 —	
T2 → ALU_A.	ADD
IR 8-0 → SE16 → 1 shift left → ALU_B	PC_en
ALU_C → PC	





BRANCH ON EQUALITY (BEQ)

S0 –

PC → Mem-Add	Mem-Read
Mem-D → IR	IR-en.
PC → ALU-A.	ADD
+2 → ALU-B	PC-en.
ALU-C → PC	

S1 –

IR_{11-9} → RF-A_1	
IR_{8-6} → RF-A_2	T_1-en
RF-D_1 → T_1	T_2-en.
RF-D_2 → T_2	

S2 –

T_1 → ALU-A.	SUB.
T_2 → ALU-B.	

If. zeroflag == 1 then S3
else S0

S3 →

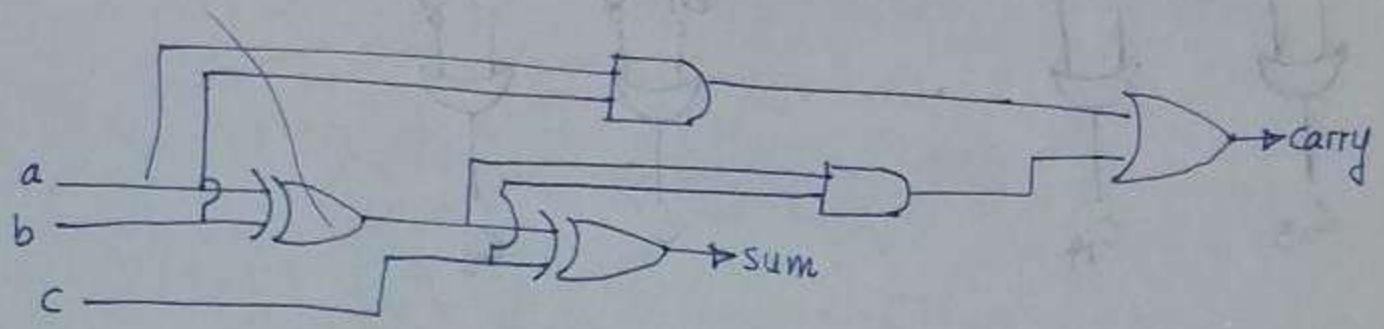
PC → ALU A.	SUB.
+2 → ALU B	T_2 – en.
ALU-C → T_2	

S4. –

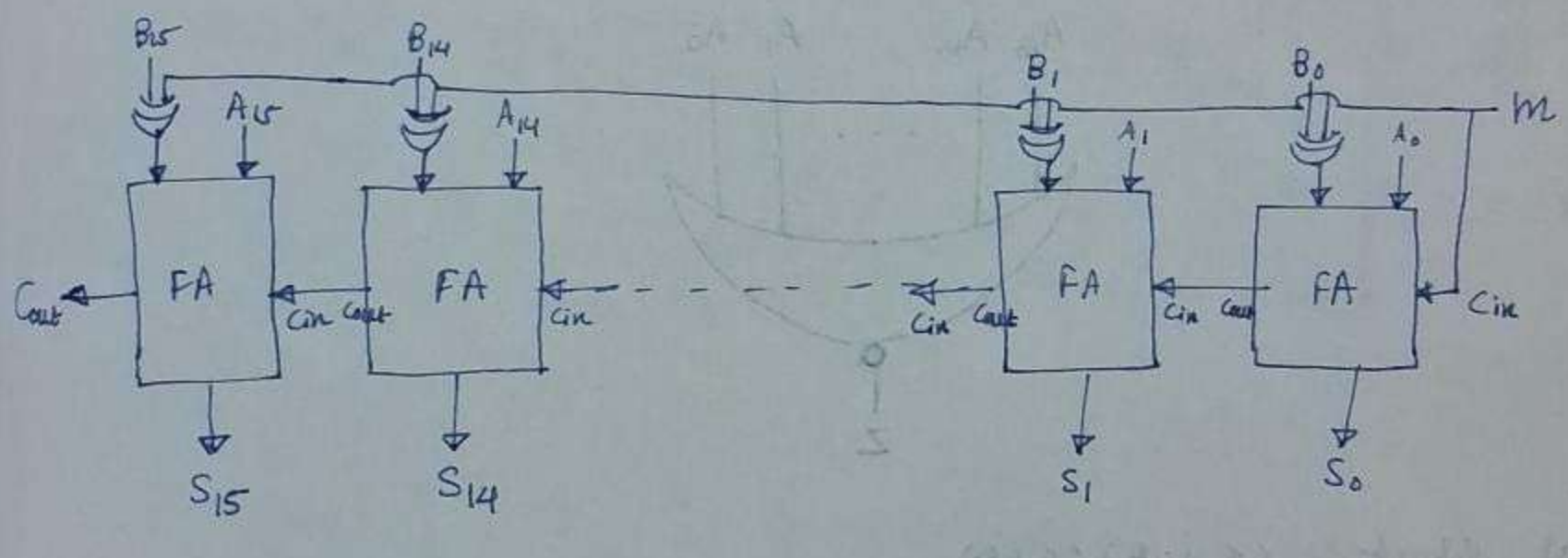
T_2 → ALU-A.	ADD.
IR_{5-0} → SE16 → 1shiftleft → ALU-B	PC-en.
ALU C → PC	

Components

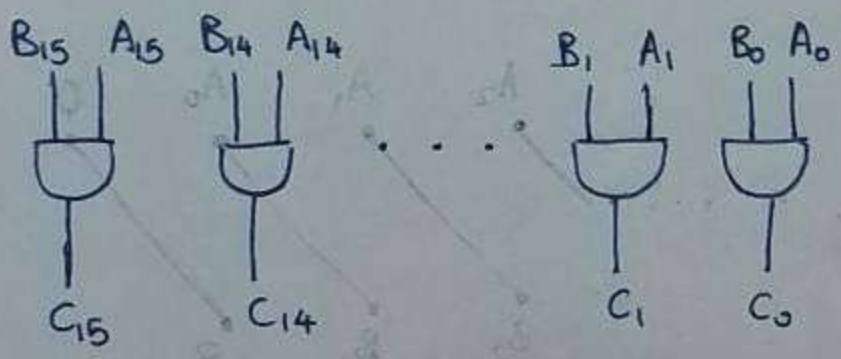
1) Full Adder :



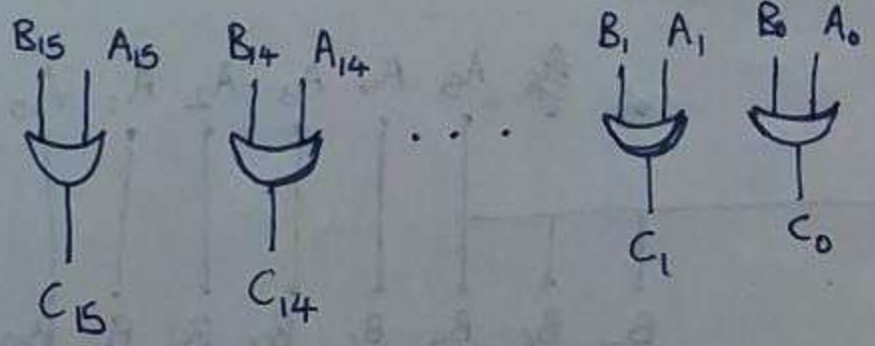
2) 16 bit adder-subtractor :



3) Bitwise AND (16 bit)

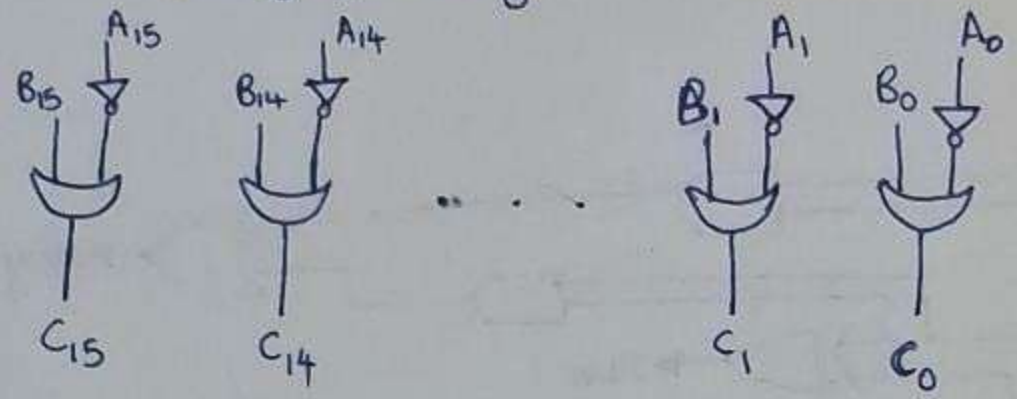


4) Bitwise OR (16 bit)

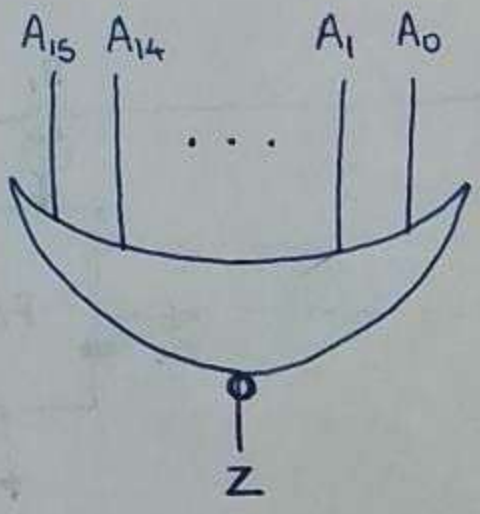


5) Bitwise IMP (16 bit)

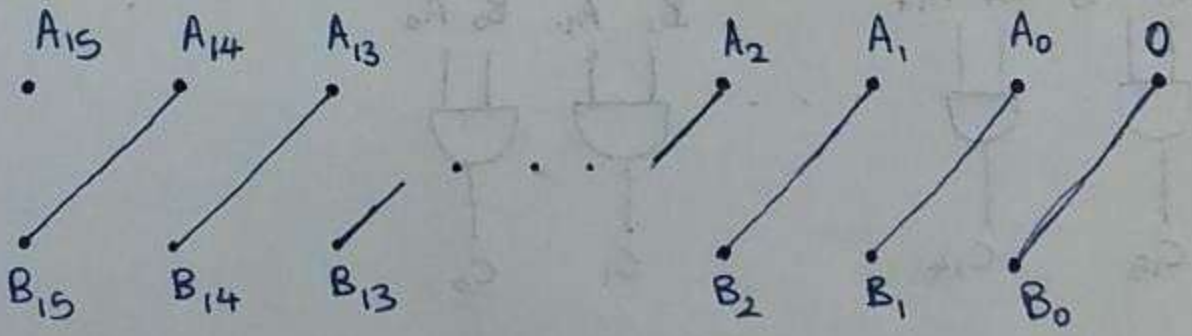
$$x \rightarrow y \equiv \bar{x} + y$$



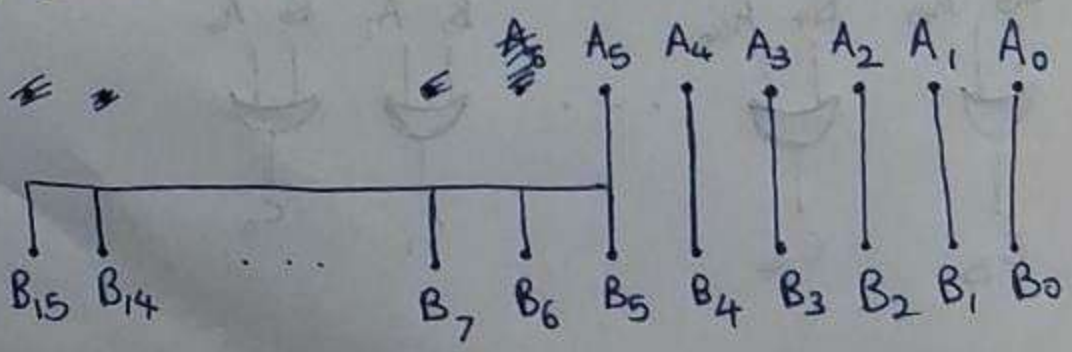
6) Zero flag / Zero detector



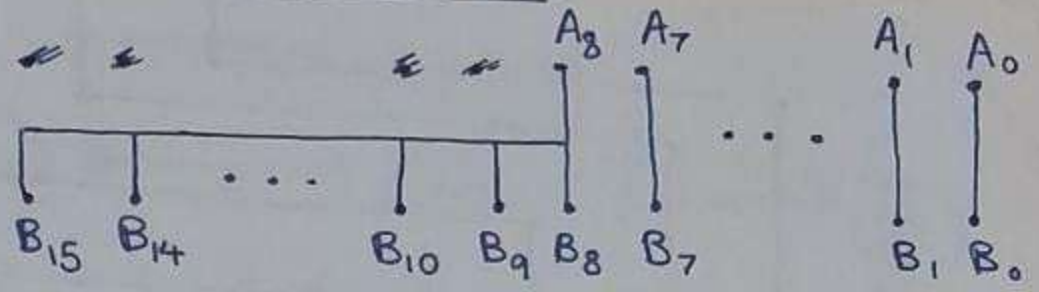
7) Shifter (1 bit) (left)



8) Sign extension (6 bit → 16 bit)

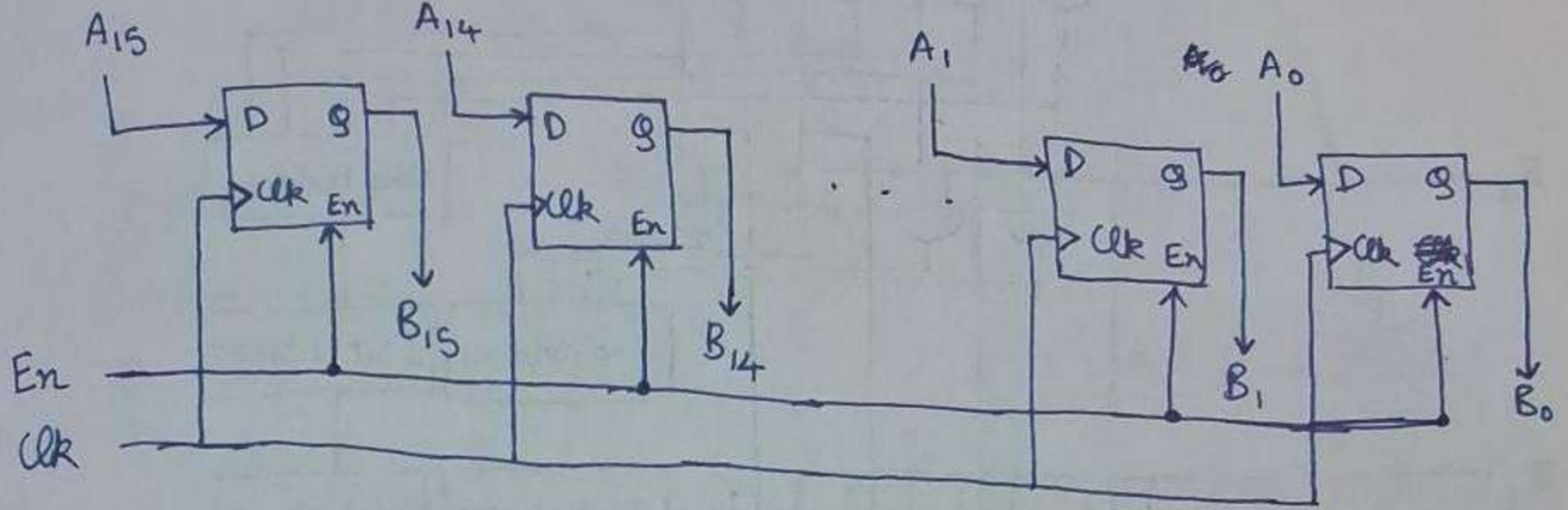


9) Sign extension (9 bit → 16 bit)

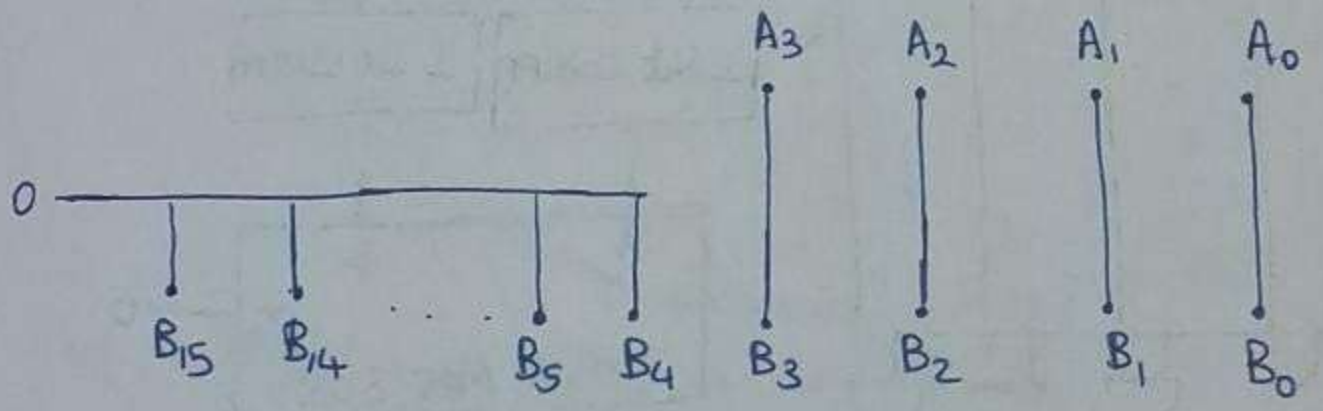


10) 16-bit Register (Parallel in, Parallel Out)

Made of 16 D-flipflops.



11) Zero Padding (4 bit → 16 bit)



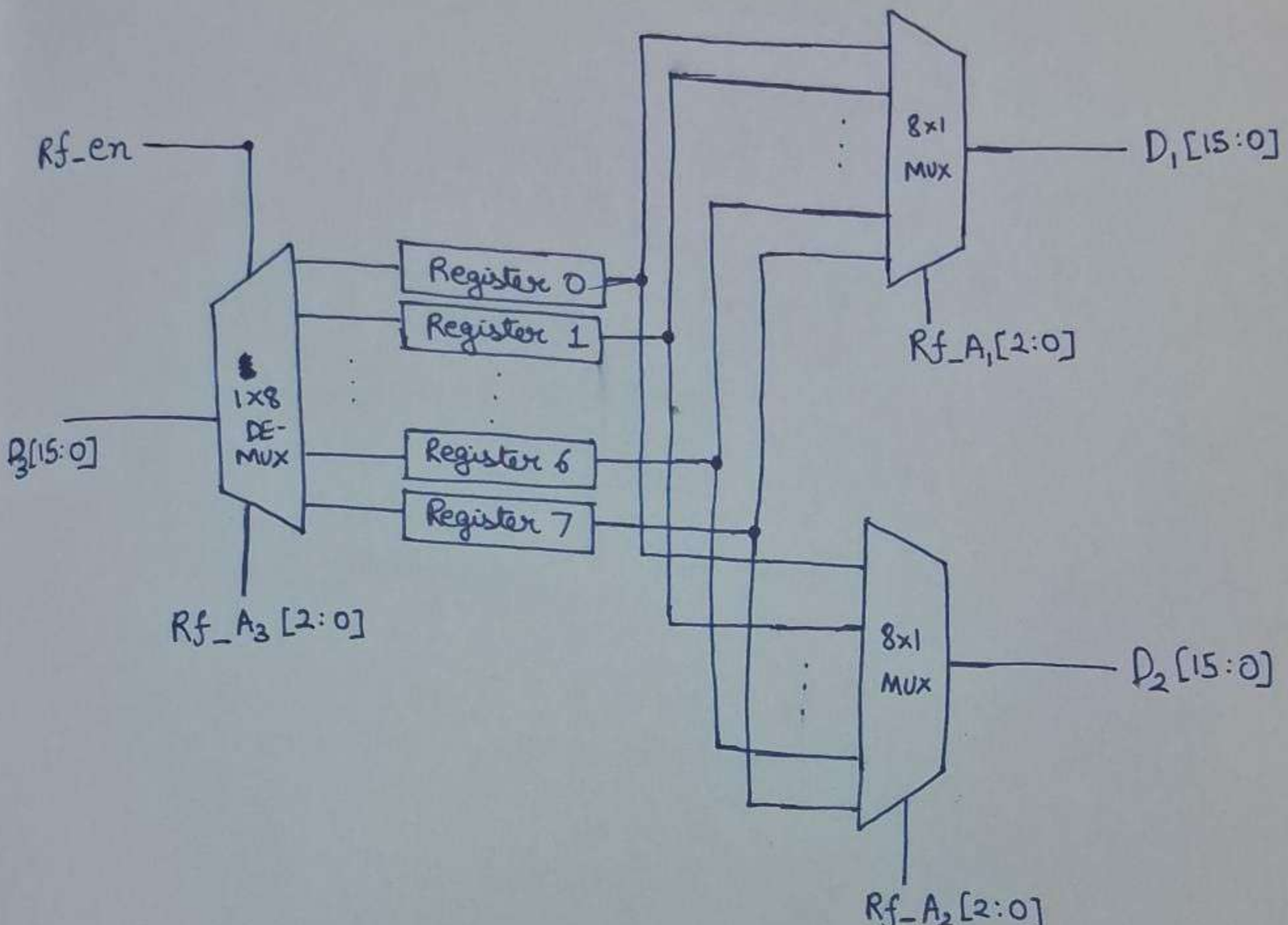
12) Multiplier (16 bit x16 bit → 16 bit)

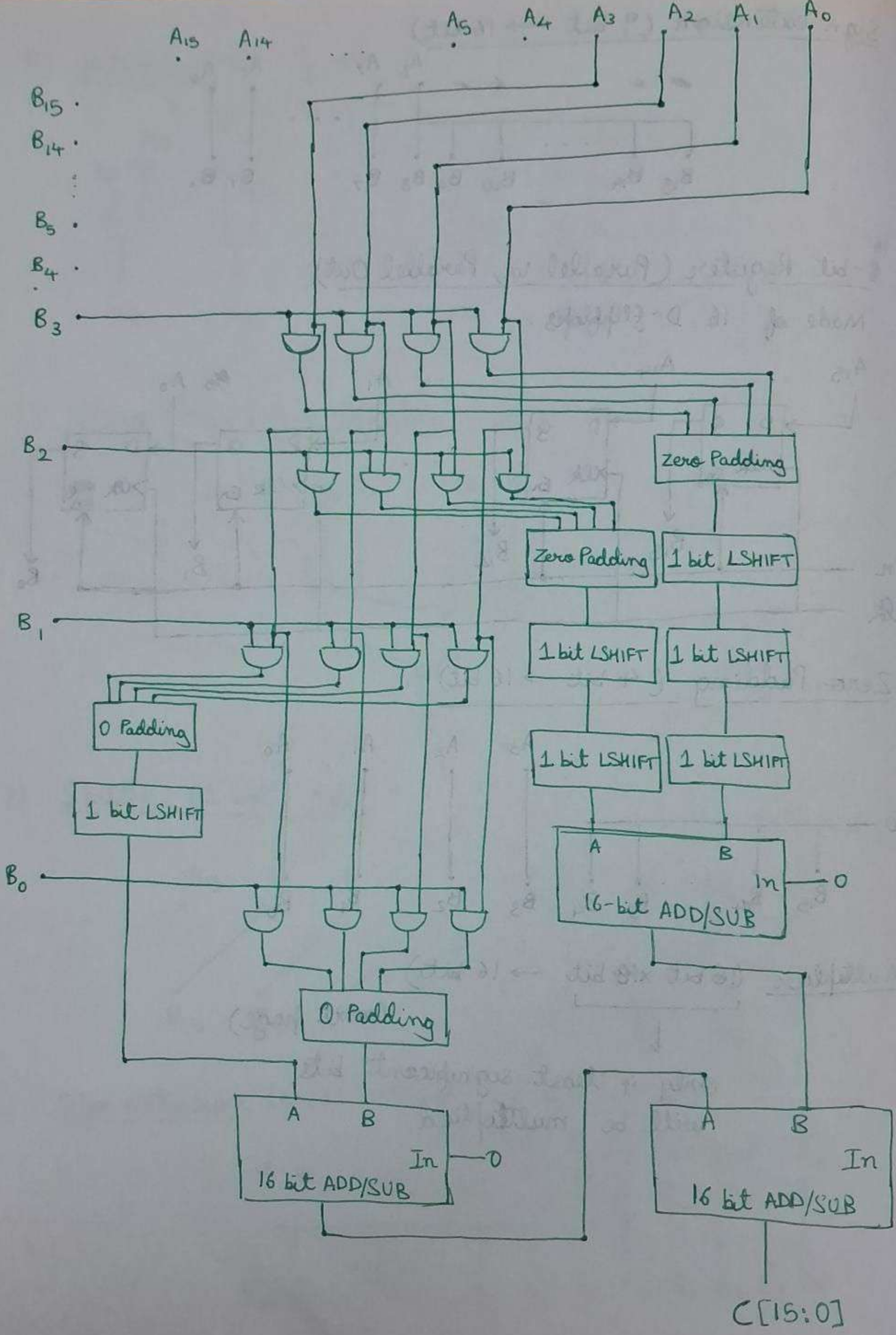
(next page)

↓

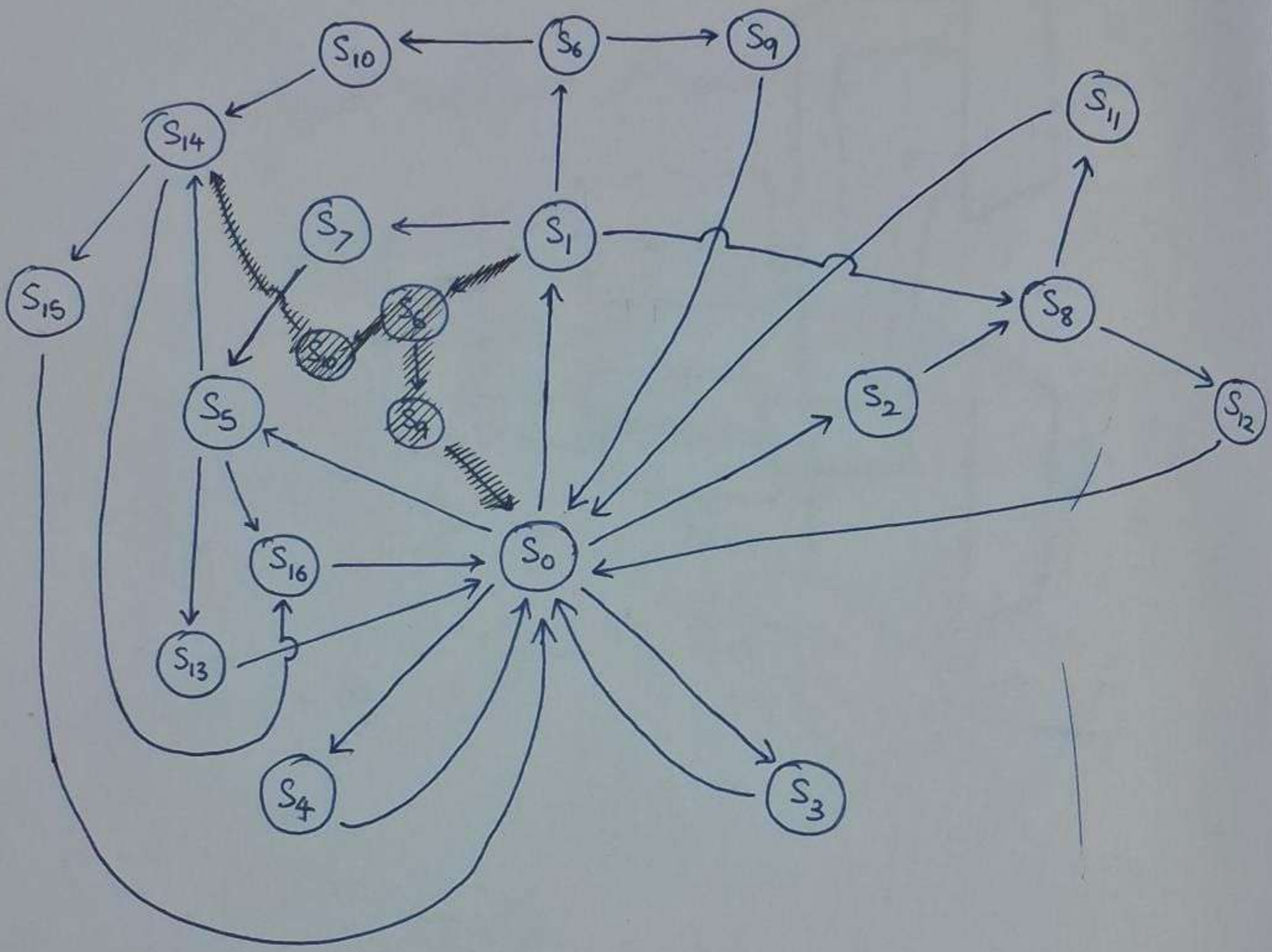
only 4 least significant bits will be multiplied

Register File (Rf_A_1, Rf_A_2, Rf_A_3, D_3, D_1, D_2, Rf-en)




A15 A14 A5 A4 A3 A2 A1 A0
B15
B14
B5
B4
B3
B2
B1
B0
Zero Padding
Zero Padding
1 bit LSHIFT
1 bit LSHIFT
1 bit LSHIFT
1 bit LSHIFT
1 bit LSHIFT
0 Padding
0 Padding
A
B
In
0
16-bit ADD/SUB
16 bit ADD/SUB
C[15:0]

State Transition Diagram



Whenever 2 there are 2 branches out of one state, we use a decoder.

Whenever there are 2 branches into one state, we use a MUX.

State Labelling

i) S_0: common to all instructions

ii) S_1: S_1 of ADD/SUB $\equiv$ S_1 of BEQ $\equiv$ S_1 of Store $\equiv$ S_1 of Load

iii) S_2: S_1 of ADI

iv) S_3: S_1 of LLI

v) S_4: S_1 of LHI

vi) S_5: S_1 of JAL $\equiv$ S_1 of J $\equiv$ S_1 of JLR $\equiv$ S_3 of BEQ [Subtracting PC by 2]

vii) S_6: S_2 of Store $\equiv$ S_2 of Load [Base + Offset*2]

viii) S_7: S_2 of BEQ

ix) S_8: S_2 of ADD $\equiv$ S_2 of ADI

x) S_9: S_3 of Store

xi) S_{10}: S_3 of Load

xii) S_{11}: S_3 of ADD

xiii) S_{12}: S_3 of ADI

xiv) S_{13}: S_4 of BEQ

xv) S_{14}: S_4 of Load $\equiv$ S_2 of JAL $\equiv$ S_2 of JLR [Writing in Reg A]

xvi) S_{15}: S_3 of JLR

xvii) S_{16}: S_2 of J $\equiv$ S_3 of JAL [Incrementing PC by Imm*2]