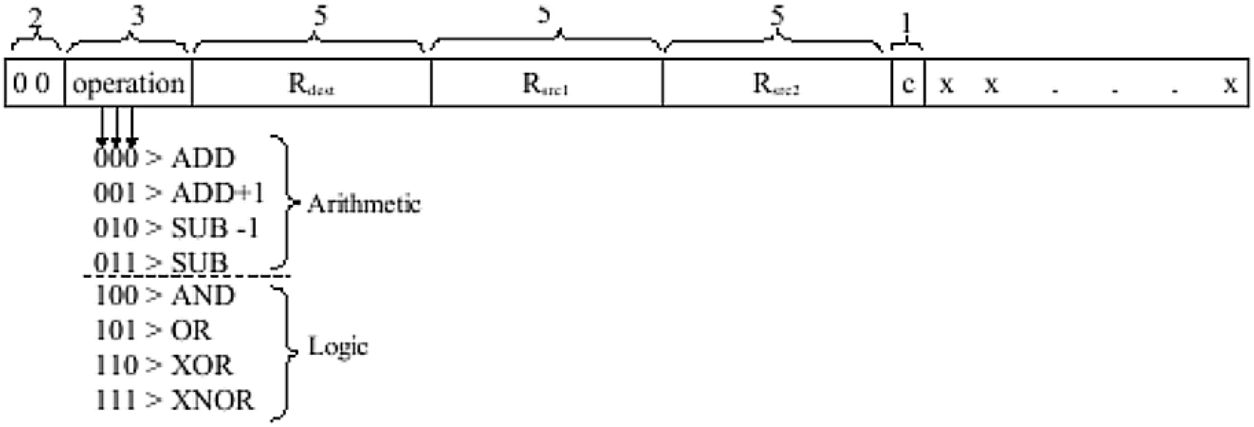
Advanced Computer Lab
Final Report

Frederic-Gerald Morcos
4-1805 – E11

ALU Instruction:



Fetch:

T0:	MAR_MUX	<-	choose [PC]
T1:	memRead	<-	1
T2:	IR	<-	MDR
	PC	<-	PC + 4

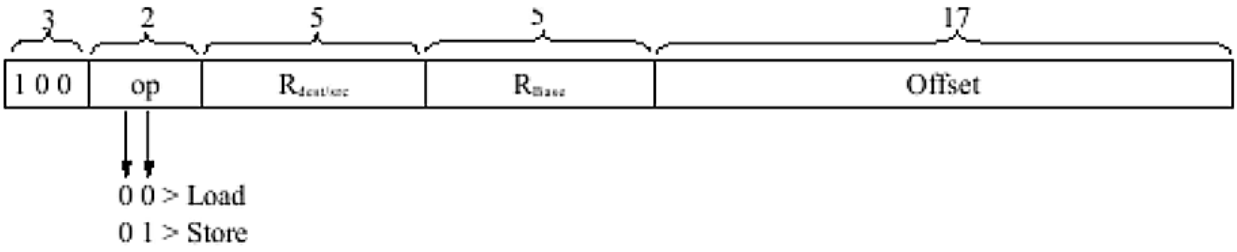
Decode:

T3:	Reg_File_address_read1	<-	IR [src1]
	Reg_File_address_read2	<-	IR [src2]
	Reg_File_address_write	<-	IR [destination]
	MUX_ALU_select	<-	choose [IR [operation]]
	MUX_ALU_b	<-	choose [RegFile_out2]

Execute:

T4:	MUX_RegFile_data	<-	choose [ALU_result]
	RegFile_write_enable	<-	1
	SetFlags	<-	IR [31]
	SetFlags	<-	IR [30]
	SetFlags	<-	IR [11]

Load/Store Instruction:



Load:
Fetch:

T0:	MAR_MUX	<-	choose [PC]
T1:	memRead	<-	1
T2:	IR	<-	MDR
	PC	<-	PC + 4

Decode:

T3:	RegFile_address_read1	<-	IR [base]
	MUX_ALU_select	<-	choose [000]
	RegFile_address_write	<-	IR [destination/source]
	MUX_ALU_b	<-	choose [IR [offset]]

Note: We need to extend the offset to 32 bits.

T4:

Load/Store	<-	ALU_Result
MUX_MAR	<-	choose [Load/Store]
memRead	<-	1
memWrite	<-	0

T5:

RegFile_write_enable	<-	1
MUX_RegFile_data	<-	MDR

Store:
Fetch:

T0:	MAR_MUX	<-	choose [PC]
T1:	memRead	<-	1
T2:	IR	<-	MDR
	PC	<-	PC + 4

Decode:

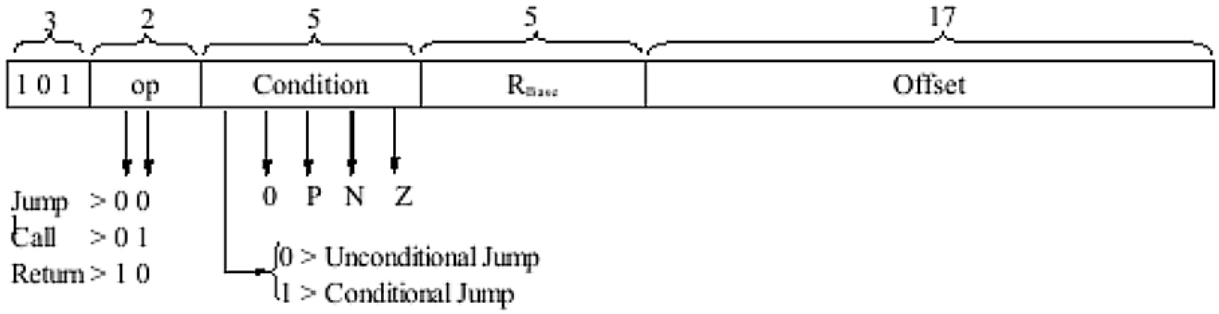
T3:	RegFile_address_read1	<-	IR [base]
	MUX_ALU_select	<-	choose [000]
	RegFile_address_read2	<-	IR [destination/source]
	MUX_ALU_b	<-	choose [IR [offset]]

Note: We need to extend the offset to 32 bits.

T4:

Load/Store	<-	ALU_Result
MUX_MAR	<-	choose [Load/Store]
memRead	<-	0
memWrite	<-	1
MDW	<-	RegFile_out2

Branch Instruction:



Jump:
Fetch:

T0:	MAR_MUX	<-	choose [PC]
T1:	memRead	<-	1
T2:	IR	<-	MDR
	PC	<-	PC + 4

Decode:

T3:	RegFile_address_read1	<-	IR [base]
	MUX_ALU_b	<-	choose [IR [offset]]
	MUX_ALU_select	<-	choose [000]

Execute:

T4:	PC	<-	ALU_Result

Call:
Fetch:

T0:	MAR_MUX	<-	choose [PC]
T1:	memRead	<-	1
T2:	IR	<-	MDR
	PC	<-	PC + 4

Decode:

T3:	RegFile_address_read1	<-	31
T4:	MAR	<-	RegFile_out2
	RegFile_write_enable	<-	1
	RegFile_Reg31	<-	RegFile_Reg31 + 4
	MUX_RegFile_data	<-	choose [ALU_Result]
	Reg_File_address_write	<-	31
	MDW	<-	PC

Execute:

T5:	Mem [MAR]	<-	MDW
	memWrite	<-	1
	PC	<-	IR [base] + IR [offset]

Return:
Fetch:

T0:	MAR_MUX	<-	choose [PC]
T1:	memRead	<-	1
T2:	IR	<-	MDR
	PC	<-	PC + 4

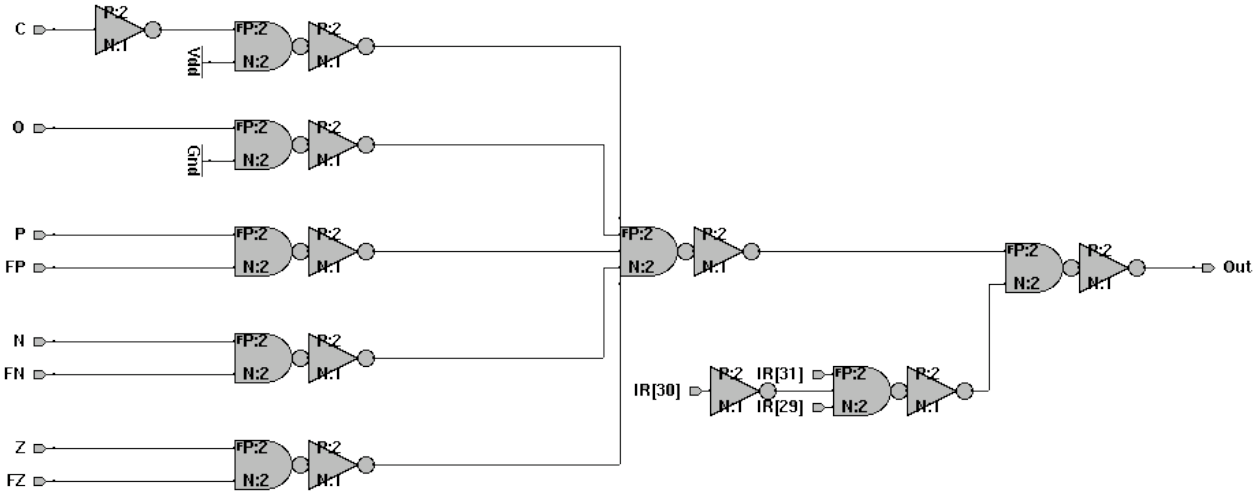
Decode:

T3:	RegFile_address_read1	<-	31
T4:	RegFile_write_enable	<-	1
	RegFile_Reg31	<-	RegFile_Reg31 - 4
	MUX_RegFile_data	<-	choose [ALU_Result]
	Reg_File_address_write	<-	31
T5:	MAR	<-	RegFile_out2 [Reg31]
	memRead	<-	1

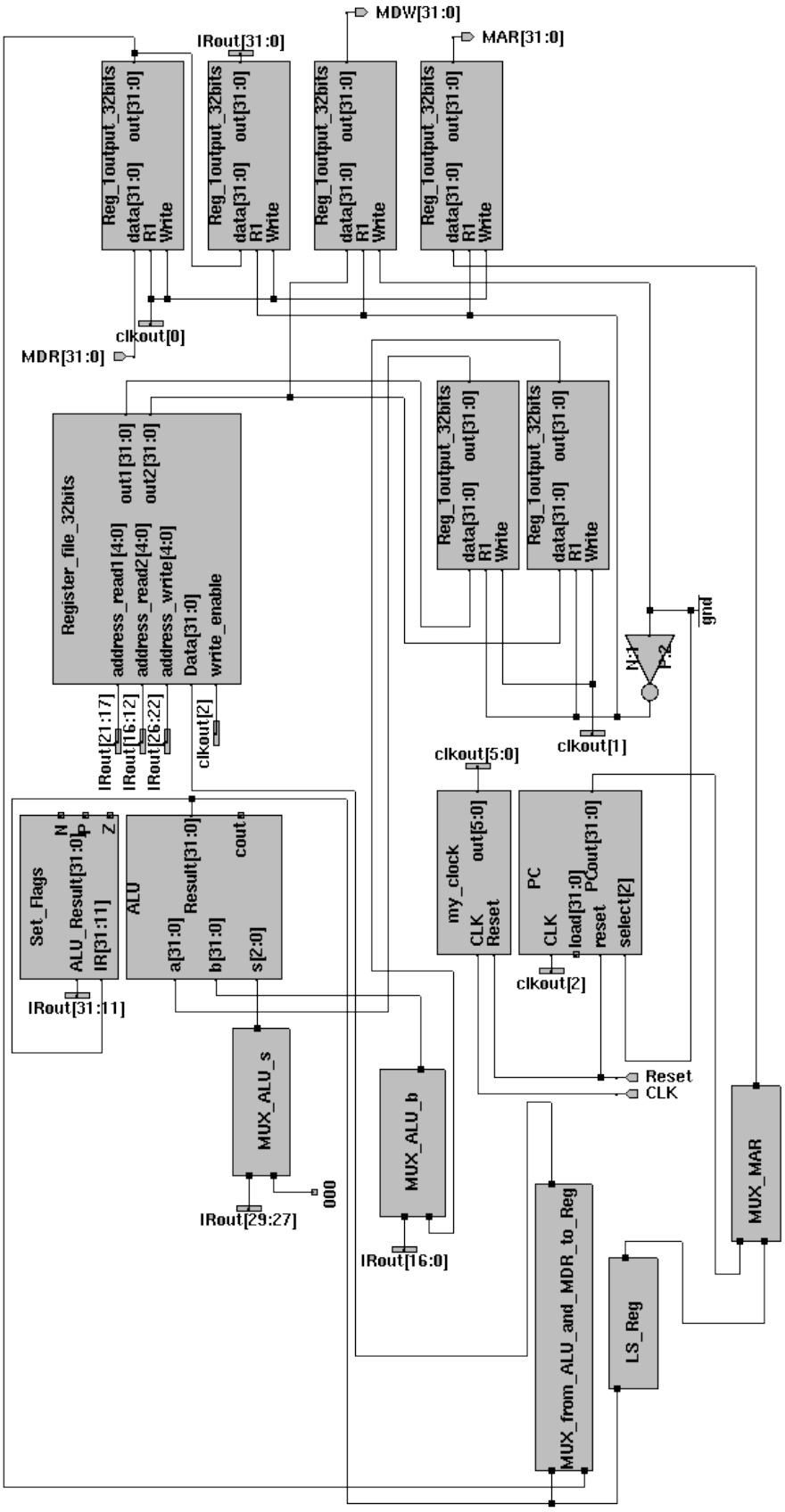
Execute:

T6:	MDR	<-	Mem [MAR]
	PC	<-	MDR

0PNZ condition checker:



Circuit (*Note: use this version to complement the other one at the end of the document*):



Logic:

Multiplexers:

MUX_ALU_b
selects RegFile_out2 when at T3 AND ALU_Result
selects 4 when at T4 AND (Call OR Return)
selects IR [offset] when at T3 AND (Load OR Store OR Jump)

MUX_ALU_s
selects 000 when at (T3 AND (Load OR Store OR Jump)) OR (T4 AND Call)
selects 011 when at T4 AND Return
selects IR [operation] when at T3 AND ALU

MUX_address_read1:
selects IR [src1] when at NOT (T3 AND (Call OR Return))
selects 31 when at T3 AND (Call OR Return)

MUX_address_read2:
selects IR [src2] when at T3 AND ALU
selects IR [destination/source] when at T3 AND Store

MUX_RegFile_Data:
selects ALU_Result when at T4 AND (ALU OR Call OR Return)
selects MDR when at T5 AND LOAD

MUX_RegFile_address_write:
selects 31 when T4 AND (Call OR Return)
selects IR [destination] when at T3 AND (ALU OR Load)

MUX_PC_Load
selects ALU_Result when at NOT (T6 AND Return)
selects MDR when at T6 AND Return

MUX_MDW
selects RegFile_out2 when at T4 AND Store
PC_out when at T5 AND Call

MUX_MAR
selects RegFile_out2 when (T4 AND Call) OR (T5 AND Return)
selects Load/Store when at T4 AND (Load OR Store)
selects PC_out when at T0

MUX_RegFile_write_enable:
1 when (T4 AND ALU) OR (T5 AND Load) OR (T4 AND (Call OR Return))

Latches:

LATCH_ALU_a
write enabled at T3

LATCH_ALU_b
write enabled at (T3 AND ALU)
OR (T3 AND (Load OR Store))
OR (T4 AND (Call OR Return))

LATCH_LOAD/STORE:
write enabled at T4 AND (Load OR Store)

Memory:

Read:	(T4 AND Load) OR T1 OR (T5 AND Return)
Write:	(T4 AND Store) OR (T6 AND Call)

Decider:

IN [1:0]	== 00	ALU Instruction
IN [4:2]	== 000	Add
	== 001	Add + 1
	== 010	Sub – 1
	== 011	Sub
	== 100	And
	== 101	Or
	== 110	Xor
	== 111	Xnor
IN [2:0]	== 100	Load/Store Instruction
IN [4:3]	== 00	Load
	== 01	Store
IN [2:0]	== 101	Branch Instruction
IN [4:3]	== 00	Jump
	== 01	Call
	== 10	Return