Computer Organization and Architecture Laboratory

Assignment 7

KGP-RISC

Group 1

Animesh Jha (19CS10070)

Nisarg Upadhyaya (19CS30031)

Instruction Format Encoding

- Common

Number of supported regs = 2^5 = 32

Arithmetic, Logical and Shift Operations				
OPCODE	Dest Reg	Source Reg	Shamt	Func
6 bits	5 bits	5 bits	10 bits (5 bits are redundant/dont cares, only the lower order 5 bits are used)	6 bits

First 6 bits will be reserved for the opcode
Next 5 bits will be reserved for the source register
Next 5 bits will be reserved for the destination register
Next 5 bits are redundant
Next 5 bits will be used for shamt
Next 6 bits will be reserved for the function name
Max shamt = 31

- Immediate Functions

Immediate Operations			
OPCODE	Dest Reg	Source Reg	Immediate
6 bits	5 bits	5 bits	16 bits

First 6 bits will be reserved for the opcode
Next 5 bits will be reserved for the source (and the destination) register
Next 5 bits will be reserved for the destination register
Next 16 bits will be reserved for the immediate val

Range of immediate: -32768 to 32767

- Branch Functions

Branch Operations			
OPCODE	Register	Destination Address	Func
6 bits	5 bits	15 bits	6 bits

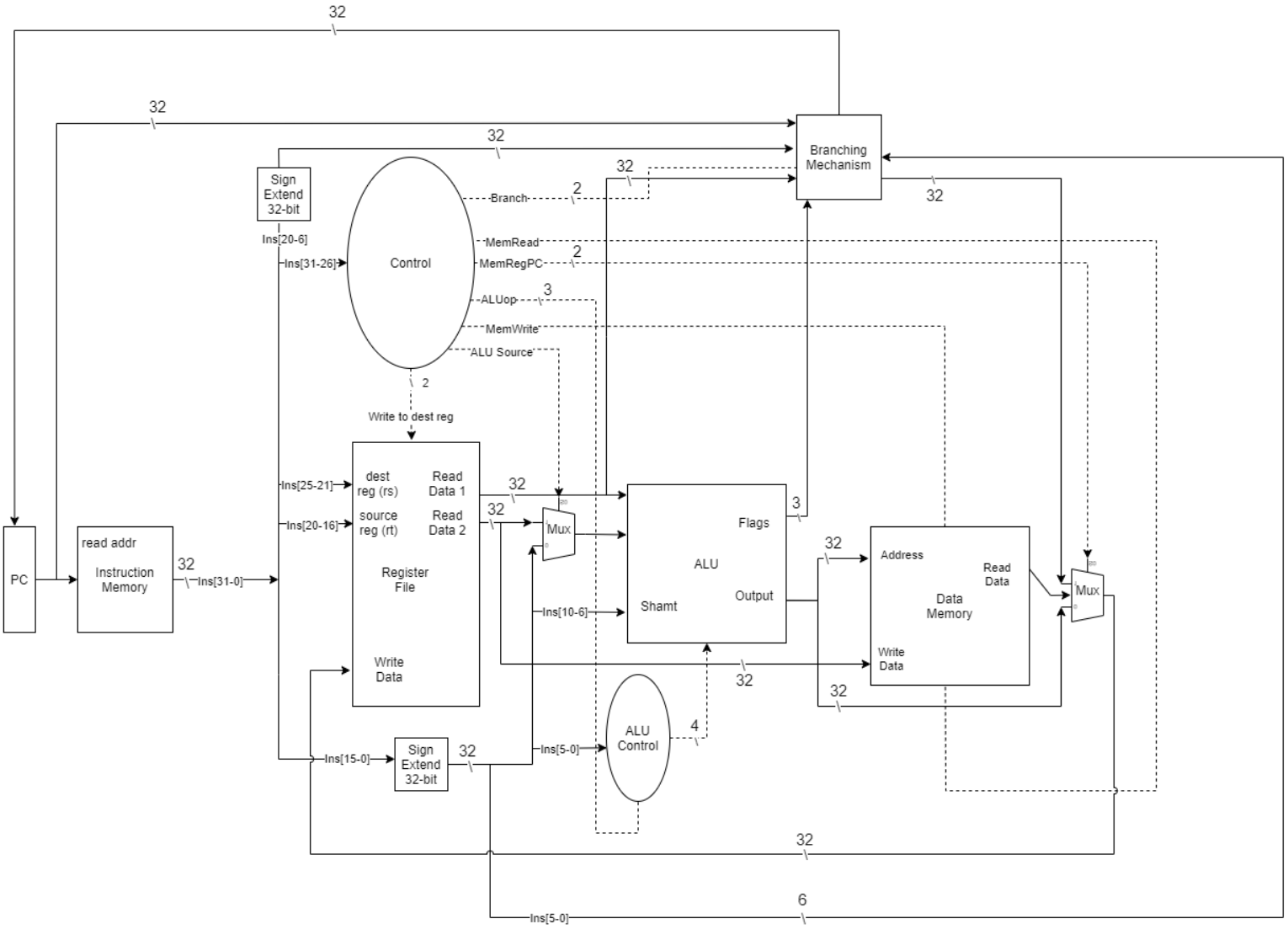
Operation	**OPCODE (in decimal)**	**Function Code (in decimal)**
Add	1	0
Compliment	1	1
AND	2	0
XOR	2	1
SLL	3	0
SRL	3	1
SLL Variable	3	2
SRL Variable	3	3
SRA	3	4
SRA Variable	3	5
Branch Register	4	0
Branch on less than 0	4	1
Branch on flag zero	4	2
Branch on flag not zero	4	3
Unconditional Branch	5	0
Branch on carry	5	1
Branch on no carry	5	2
Branch and link	6	0
Add Immediate	60	NA
Compliment Immediate	61	NA
Load Word	62	NA
Store Word	63	NA

Control Unit

Takes as input 6 bit OPCODE and assigns values to the different control lines. See diagram on next page.

OPCODE	ALUop	ALUsource	WriteReg	MemWrite	MemRead*	MemRegPC	Branch
000001	001	1	10	0	0	00	00
000010	010	1	10	0	0	00	00
000011	011	1	10	0	0	00	00
000100	000	1	00	0	0	00	01
000101	000	1	00	0	0	00	10
000110	000	1	01	0	0	10	11
111100	101	0	10	0	0	00	00
111101	110	0	10	0	0	00	00
111110	101	0	11	0	1	01	00
111111	101	0	00	1	1	00	00

ALUop - This combined with the function code lets the ALU control decide what operation to choose.
ALUsource - Immediate or register for second operand of ALU.
WriteReg[1:0] - WriteReg[1] tells whether to write or not, WriteReg[0] indicates whether to write to (rs) or (rt), if WriteReg[1] is 0 and WriteReg[0] is 1 it is a special case which indicates to write to reg 31.
MemWrite and MemRead - Whether to read/write from memory or not.
*MemRead actually corresponds to the enable of the BRAM in verilog, hence it has to be kept 1 for both the read and write operation
MemRegPC - What to write to the register, the ALU result, the memory read, or the value of program counter + 4.
Branch - Whether branch has been enabled or not. If 00 it has not been enabled. Else 01 corresponds to branches using opcode 4, 10 corresponds to branches using opcode 5 and 11 corresponds to branches using opcode 6.



The dotted lines are the control lines.
The solid lines are the data lines.
Unless explicitly shown line width is one.

ALU (opcodes 1,2,3,60,61)

INPUT

- 32 bit reg 1
- 32 bit reg 2
- 5 bit shift amount
- Control Signal (encoding bits given on next page)

Output

- 32 bit reg
- 3 bit flag [zero, negative, carry]

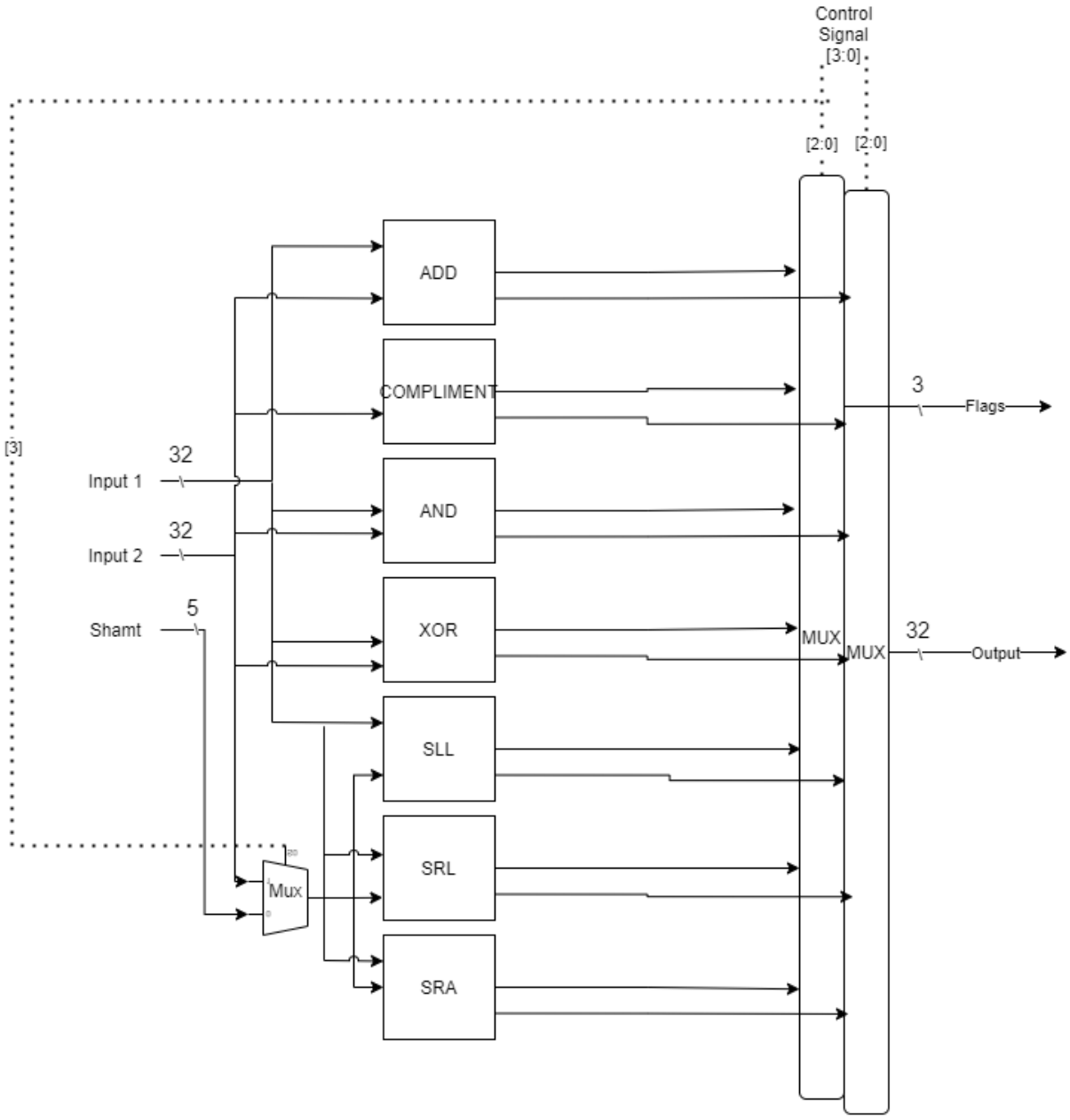
Use mux to decide whether to take input at reg2 from register or from immediate and sign extend to change immediate 16 bit to 32 bit, and feed into reg2. Additionally one bit of the control signal also decides whether to take the shift amount from reg2 or the passed shift amount.

Internal support ==, != , < for branching (opcode 4)

ALU Control

Gets input from the main control unit and the function code and decides the necessary operation for the ALU. Generates a 4 bit control signal (explained on the next page).

ALUop (from control unit)	Function Code	Control Signal
000	x	0111
001	000000	0000
001	000001	0001
010	000000	0010
010	000001	0011
011	000000	0100
011	000001	0101
011	000010	1100
011	000011	1101
011	000100	0110
011	000101	1110
101	x	0000
110	x	0001



ControlSignal[3] - if 1 select input2 else select shamt
ControlSignal[2:0]

- if 000 then add
- if 001 then compliment
- if 010 then and
- if 011 then xor
- if 100 then sll
- if 101 then srl
- if 110 then sra
- if 111 then reset output and flags to 0