Indian Institute of Technology, Kharagpur
Department of Computer Science and Engineering

CS39001: Computer Organisation Lab

ASSIGNMENT 6: KGP-RISC PROCESSOR

6th November, 2022

Group Number: 69

Jatin Gupta
20CS10087

Rushil Venkateswar
20CS30045

Contents

1. Instruction Set Architecture

Class	Instruction	Usage	Meaning
Arithmetic	Add	add rs,rt	$rs \leftarrow (rs) + (rt)$
	Comp	comp rs,rt	$rs \leftarrow$ 2's Complement (rt)
Logic	AND	and rs,rt	$rs \leftarrow (rs) \wedge (rt)$
	XOR	xor rs,rt	$rs \leftarrow (rs) \oplus (rt)$
Shift	Shift left logical	shll rs, sh	$rs \leftarrow (rs)$ left-shifted by sh
	Shift right logical	shrl rs, sh	$rs \leftarrow (rs)$ right-shifted by sh
	Shift left logical variable	shllv rs, rt	$rs \leftarrow (rs)$ left-shifted by (rt)
	Shift right logical variable	shrlv rs, rt	$rs \leftarrow (rs)$ right-shifted by (rt)
	Shift right arithmetic	shra rs, sh	$rs \leftarrow (rs)$ arithmetic right-shifted by sh
	Shift right arithmetic variable	shrav rs, rt	$rs \leftarrow (rs)$ arithmetic right-shifted by (rt)
Complex	Diff	diff rs, rt	$rs \leftarrow$ the LSB bit at which rs and rt differ
Arithmetic Immediate	Add immediate	addi rs,imm	$rs \leftarrow (rs) + imm$
	Complement Immediate	compi rs,imm	$rs \leftarrow$ 2's Complement (imm)
Memory	Load Word	lw rt, imm(rs)	$rt \leftarrow mem[(rs) + imm]$
	Store Word	sw rt, imm(rs)	$mem[(rs) + imm] \leftarrow (rt)$
Branch	Branch Register	br rs	goto (rs)
	Branch on less than 0	bltz rs, L	if(rs) $<$ 0 then goto L
	Branch on flag zero	bz rs, L	if(rs) $=$ 0 then goto L
	Branch on flag not zero	bnz rs, L	if$(rs) \neq 0$ then goto L
	Unconditional branch	b L	goto L
	Branch and link	bl L	goto L; \$ra $\leftarrow (PC) + 4$
	Branch on Carry	bcy L	goto L if Carry $= 1$
	Branch on No Carry	bncy L	goto L if Carry $= 0$

2. Description of Instruction Format

2.1 R-Format Instructions

opcode	rs	rt	shamt	Don't care	func
6 bits	5 bits	5 bits	5 bits	6 bits	5 bits

Instruction	opcode	func
add	000000	00000
comp	000000	00001
and	000001	00000
xor	000001	00001
shll	000010	00000
shrl	000010	00001
shllv	000010	00010
shrlv	000010	00011
shra	000010	00100
shrav	000010	00101
diff	000011	00000

2.2 I-Format, Load/Store, Jump Instructions

opcode	rs	Don't care	imm
6 bits	5 bits	5 bits	16 bits

Instruction	opcode	func
addi	000100	N.A.
compi	000101	N.A.

opcode	rs	rt	imm
6 bits	5 bits	5 bits	16 bits

Instruction	opcode	func
lw	000110	N.A.
sw	000111	N.A.

opcode	rs	Don't care	L
6 bits	5 bits	5 bits	16 bits

Instruction	opcode	func
bltz	001000	N.A.
bz	001001	N.A.
bnz	001010	N.A.

2.3 J-Format Instruction (With Register Arguments)

opcode	rs	Don't care
6 bits	5 bits	21 bits

Instruction	opcode	func
br	001011	N.A.

2.4 J-Format Instructions (Without Register Arguments)

opcode	L
6 bits	26 bits

Instruction	opcode	func
b	001100	N.A.
bl	001101	N.A.
bcy	001110	N.A.
bncy	001111	N.A.

3. Register Usage Convention

The register file contains 32 registers each of size 32-bits.

Register	Function	Register Number	Register Code
$zero	Zero Register, stores the value 0 always	0	00000
$pc	Program Counter, points to next instructions to be executed	1	00001
$v0-$v1	Return values from subroutines	$2-3$	$00010-00011$
$a0-$a3	Arguments to subroutines	$4-7$	$00100-00111$
$t0-$t11	Temporary Registers	$8-19$	$01000-10011$
$s0-$s9	Saved Registers	$20-29$	$10100-11101$
$sp	Stack Pointer	30	11110
$ra	Return Address	31	11111

4. Architecture of the ALU

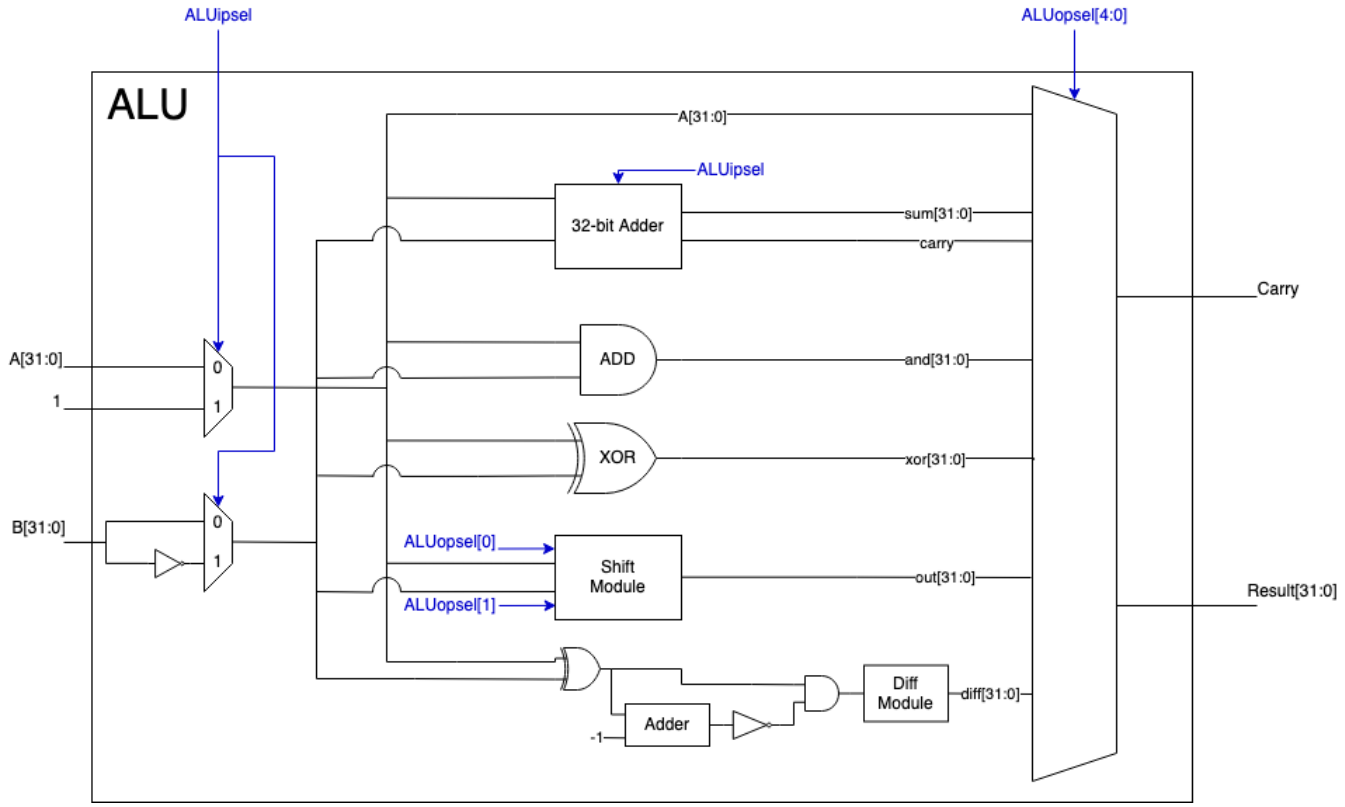

Figure 1: Arithmetic and Logic Unit

4.1 Explanation of diff command

Initially, we find the XOR of A[31:0] and B[31:0]. The result (= C[31:0]) is used to form $X = C \wedge \neg(C-1)$. X is of the form 2^i where i ranges from $0-31$ and we use switch case to output i for each 2^i.

5. Architecture of the Processor/Datapath

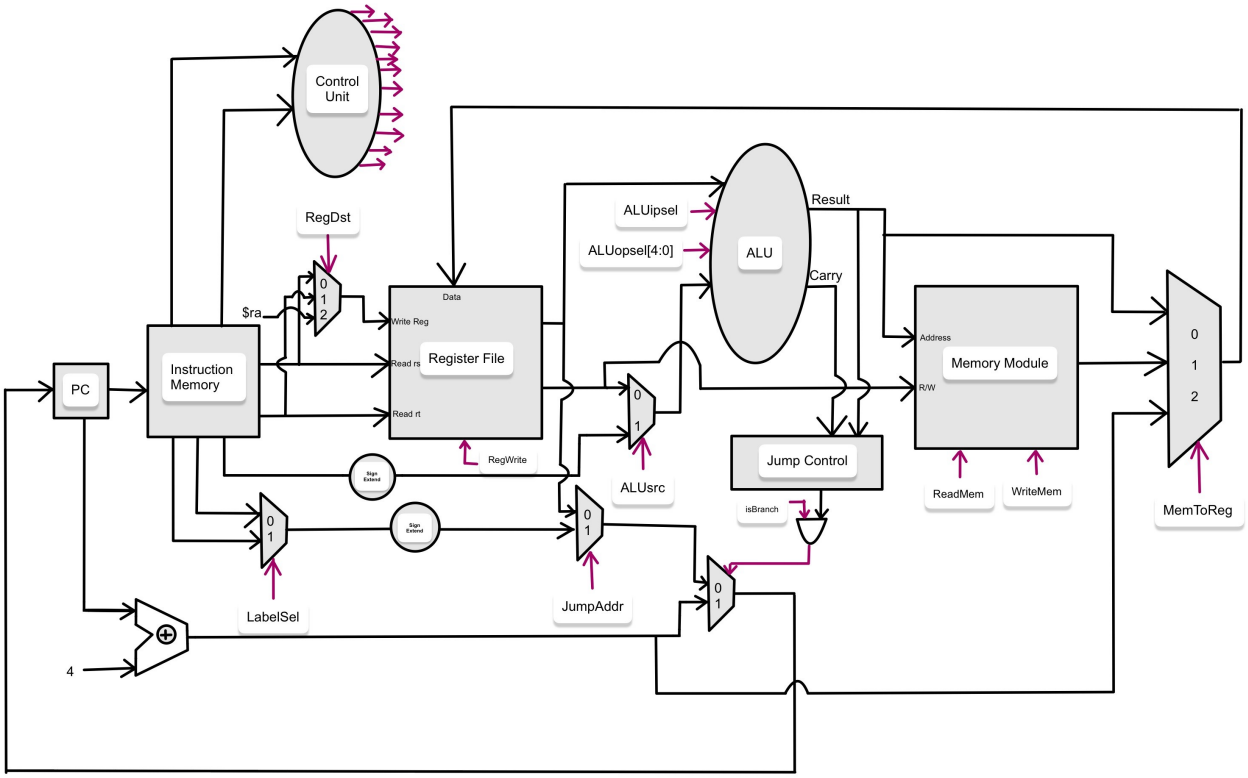

Figure 2: Datapath

6. Truth Table for Control Signals

Instr	opcode	func	Reg Dst	Reg Write	Mem Read	Mem Write	Mem To Reg	ALU src	ALU opsel	ALU ipsel	is Branch	Jump Addr	Label Sel
add	000000	00000	00	1	0	0	10	0	00001	0	0	X	X
comp	000000	00001	00	1	0	0	10	0	00010	1	0	X	X
and	000001	00000	00	1	0	0	10	0	00011	0	0	X	X
xor	000001	00001	00	1	0	0	10	0	00100	0	0	X	X
shll	000010	00000	00	1	0	0	10	1	01010	0	0	X	X
shrl	000010	00001	00	1	0	0	10	1	01000	0	0	X	X
shllv	000010	00010	00	1	0	0	10	0	01010	0	0	X	X
shrlv	000010	00011	00	1	0	0	10	0	01000	0	0	X	X
shra	000010	00100	00	1	0	0	10	1	01001	0	0	X	X
shrav	000010	00101	00	1	0	0	10	0	01001	0	0	X	X
diff	000011	00000	00	1	0	0	10	0	00101	0	0	X	X
addi	000100	N.A.	00	1	0	0	10	1	00001	0	0	X	X
compi	000101	N.A.	00	1	0	0	10	1	00010	1	0	X	X
lw	000110	N.A.	01	1	1	0	01	1	00110	0	0	X	X
sw	000111	N.A.	XX	0	0	1	XX	1	00110	0	0	X	X
bltz	001000	N.A.	XX	0	0	0	XX	X	00000	X	1	0	1
bz	001001	N.A.	XX	0	0	0	XX	X	00000	X	1	0	1
bnz	001010	N.A.	XX	0	0	0	XX	X	00000	X	1	0	1
br	001011	N.A.	XX	0	0	0	XX	X	00000	X	1	1	X
b	001100	N.A.	XX	0	0	0	XX	X	00000	X	1	0	0
bl	001101	N.A.	10	1	0	0	00	X	00000	X	1	0	0
bcy	001110	N.A.	XX	0	0	0	XX	X	00000	X	1	0	0
bncy	001111	N.A.	XX	0	0	0	XX	X	00000	X	1	0	0

Explanation of control signals:

1. **opcode** : This tells us which kind of instruction is to be executed (Operation Code).
2. **func** : This tells us which function is to be executed based on the opcode.
3. **RegDst** : Specifies destination register (rs, rt or ra).
4. **RegWrite** : Specifies whether to write to a register or not.
5. **MemRead** : Specifies whether to read from memory or not.
6. **MemWrite** : Specifies whether to write to memory or not.
7. **MemToReg** : Specifies which line to select to write data to register from (PC+4, Mem output or ALU result).
8. **ALUsrc** : Select one of rt (for R-format instruction) or sign extended value (for I-format instruction).
9. **ALUopsel** : Select which ALU operation to perform. ALUopsel[3] is 1 for shift instructions and 0 otherwise. Other bits specify which type of instruction is executed.
10. **ALUipSel** : Specifies if the instruction requires 2's complement or not (for 1, the 2's complement of B is found).
11. **isBranch** : Specifies whether a jump instruction is encountered or not.
12. **JumpAddr** : Specifies whether to take the jump address from register or from instruction.
13. **LabelSel** : Specifies whether to use 26-bit label or 16-bit label.

7. Truth Table for Jump Control

Instr	opcode	zero	sign	carry	validJump
bltz	001000	0	1	X	1
bz	001001	1	0	X	1
bnz	001010	0	X	X	1
br	001011	X	X	X	1
b	001100	X	X	X	1
bl	001101	X	X	X	1
bcy	001110	X	X	1	1
bncy	001111	X	X	0	1

X denotes don't care values.
Only under these conditions is validJump equal to 1, otherwise it is equal to 0.