Computer Organization and Architecture Laboratory

Group 23

Sampreeth R S (21CS30038)
Yash Kumar (21CS30059)

1. Instruction format and Encoding

1.1 R type instructions:

Opcode	Source 1	Source 2	Destination	Shift Amount	Function code
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Instruction	Usage	Opcode	Funct code
Add	add rd,rs,rt	000000	000001
Subtract	sub rd,rs,rt	000000	000010
And	and rd,rs,rt	000000	000011
Or	or rd,rs,rt	000000	000100
Xor	xor rd,rs,rt	000000	000101
Not	not rd,rs,rt	000000	000110
Shift left	sll rd,rs	000000	000111
Shift right logical	srl rd,rs	000000	001000
Shift right arithmetic	sra rd,rs	000000	001001
Move	move rd,rs	000000	001010
Push	push rd	000000	001011
Pop	pop rd	000000	001100
Return	ret	000000	001101

1.2 I type instructions:

Opcode	Source	Destination	Immediate Value
6 bits	5 bits	5 bits	16 bits

Instruction	Usage	Opcode
AddI	addi rs, imm	000001
SubtractI	subi rs, imm	000010
AndI	andi rs, imm	000011
OrI	ori rs, imm	000100
XorI	xori rs, imm	000101
NotI	noti rs, imm	000110
Shift Left Immediate	slai rs, imm	000111
Shift Right Logical Immediate	srli rs, imm	001000
Shift Right Arithmetic Immediate	srai rs, imm	001001
Branch	br imm	001010
Branch if Minus	bmi rs, imm	001011
Branch if Plus	bpl rs, imm	001100
Branch if equal to zero	bz rs, imm	001101
Load	Ld rd, rs, imm	001110
Store	st rd, rs, imm	001111
Load Stack Pointer	ldsp sp, rs, imm	010000
Store Stack Pointer	stsp sp, rs, imm	010001
Call	call imm	010010

2.3 Miscellaneous Instructions:

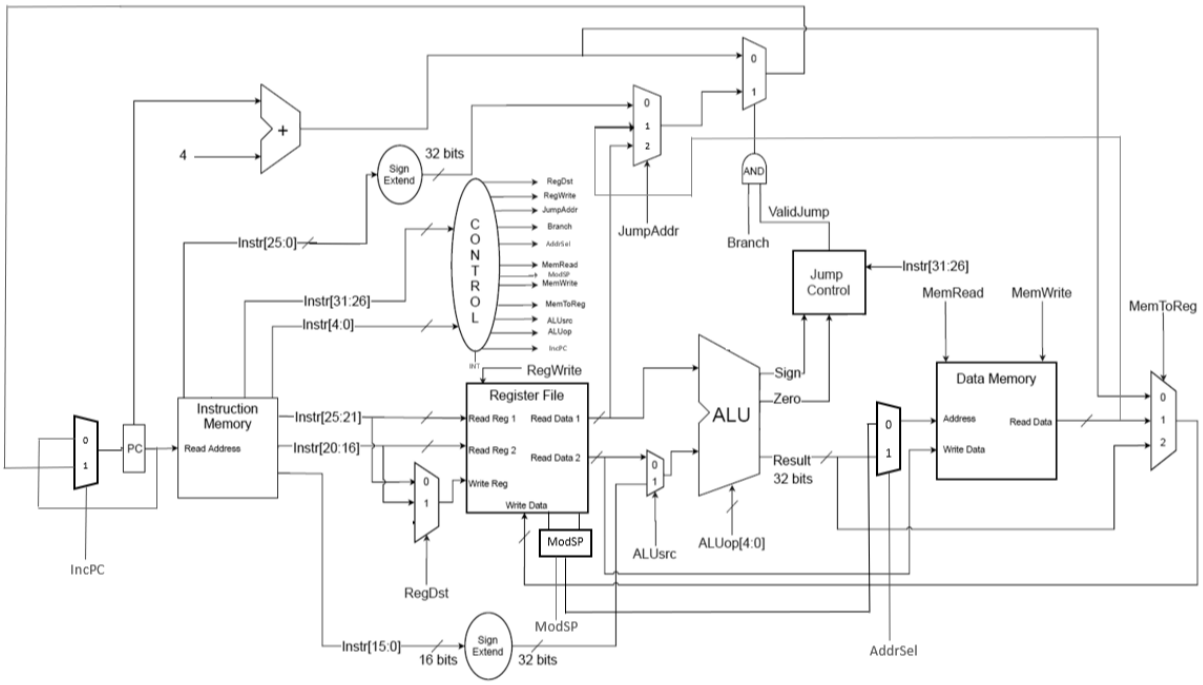
Opcode	Don't care
6 bits	26 bits

Instruction	Usage	Opcode
Halt	halt	111110
No operation	nop	111111

2. Register Usage Convention

Register	Function	Register Number
$zero	Hardwired to 0	0
$r1-$r15	Temporary registers	1-15
$sp	Stack Pointer	16
$pc	Programme counter	17

3.Datapath



4. Control Unit Design

Instr	opcode	func	Reg dest	Reg write	Mem Read	Mem Write	Mem To Reg	ALU src	ALU op	Branch	JumpAddr	Addr Sel	Mod SP	Inc PC
Add	000000	000001	1	1	0	0	2	0	0	0	X	X	0	1
Subtract	000000	000010	1	1	0	0	2	0	1	0	X	X	0	1
And	000000	000011	1	1	0	0	2	0	5	0	X	X	0	1
Or	000000	000100	1	1	0	0	2	0	7	0	X	X	0	1
Xor	000000	000101	1	1	0	0	2	0	8	0	X	X	0	1
Not	000000	000110	1	1	0	0	2	0	6	0	X	X	0	1
Shift left	000000	000111	1	1	0	0	2	0	3	0	X	X	0	1
Shift right logical	000000	001000	1	1	0	0	2	0	4	0	X	X	0	1
Shift right arithmetic	000000	001001	1	1	0	0	2	0	2	0	X	X	0	1
Move	000000	001010	1	1	0	0	2	0	9	0	X	X	0	1
Push	000000	001011	X	0	0	1	X	X	9	0	X	0	2	1
Pop	000000	001100	1	1	1	0	1	X	9	0	X	0	1	1
Return	000000	001101	X	0	1	1	2	X	9	0	1	0	1	1
AddI	000001	X	0	1	0	0	2	1	0	0	X	X	0	1
SubtractI	000010	X	0	1	0	0	2	1	1	0	X	X	0	1
AndI	000011	X	0	1	0	0	2	1	5	0	X	X	0	1

Orl	000100	X	0	1	0	0	2	1	7	0	X	X	0	1
Xorl	000101	X	0	1	0	0	2	1	8	0	X	X	0	1
Notl	000110	X	0	1	0	0	2	1	6	0	X	X	0	1
Shift Left Immediate	000111	X	0	1	0	0	2	1	3	0	X	X	0	1
Shift Right Logical Immediate	001000	X	0	1	0	0	2	1	4	0	X	X	0	1
Shift Right Arithmetic Immediate	001001	X	0	1	0	0	2	1	2	0	X	X	0	1
Branch	001010	X	X	0	0	0	X	1	0	1	0	X	0	1
Branch if Minus	001011	X	X	0	0	0	X	1	0	1	0	X	0	1
Branch if Plus	001100	X	X	0	0	0	X	1	0	1	0	X	0	1
Branch if equal to zero	001101	X	X	0	0	0	X	1	0	1	0	X	0	1
Load	001110	X	0	1	1	0	1	1	0	0	X	1	0	1
Store	001111	X	X	0	0	1	X	1	0	0	X	1	0	1
Load Stack Pointer	010000	X	0	1	1	0	1	1	0	0	X	1	0	1
Store Stack Pointer	010001	X	0	0	0	1	X	1	0	0	X	1	0	1
Call	010010	X	X	0	0	0	X	1	0	0	X	0	2	1
Halt	111110	X	X	0	0	0	X	X	10	0	X	X	0	0
No Operation	111111	X	X	0	0	0	X	X	10	0	X	X	0	1

Description of Control Signals:

1) **RegDst:** Selects the address of the destination register from rs or rt.
2) **RegWrite:** Activates the write port of the register file.
3) **MemRead:** Triggers a read operation.
4) **MemWrite**: Triggers a write operation.
5) **MemToReg:** Select which value to write onto the register
6) **ALUsrc**: 0 for R-type instructions and 1 for I-type instructions. To control the 2nd input of the ALU.
7) **ALUop**: Determines the type of ALU operation. (Example: 0 is for add, 1 is for subtract, 2 is for shift right arithmetic, etc.)
8) **Branch**: Determines whether an instruction is a branch instruction or not
9) **JumpAddr**: Determines the kind of address to be selected during a jump operation. 1 is for stack pointer, 2 is for register value address and 0 is for immediate value address.
10) **AddrSel:** Selects one of ALUout and stack pointer value for the memory address.
11) **Mod SP**: Determines how the Stack Pointer is modified. 0 is when the stack pointer is not modified, 1 is when the stack pointer is increased by 4 and 2 is when the stack pointer is decreased by 4.
12) **Inc PC**: Determines whether or not PC will be incremented. PC is incremented when Inc PC is 1 and not incremented when it is 0.