Cs 264 Computer Architecture Lab

Reduced Instruction Set Architecture

230001006 Aman Gupta 230001068 Rayavarapu Sreechand

Instructions in ISA have different format based on their instruction type. Instructions consists of different fields like Op code, Rs, Rt, Rd, Sa, Function, Immediate. Reducing instruction set architecture for given functions will require reduction in number of bits for each field of instructions.

<u>Reduced ISA</u>

Total number of Instructions :	**31 instructions**	**Number of bits for each field of instructions**	
• *R type instructions :*	14	op - 5, rs, rt, rd, sa – 5 for each , func-6	=31bits
• *I type instructions :*	16	op - 5, rs, rt – 5 for each , imm - 16	=31bits
• *J type instructions:*	1	op - 5, imm - 26	=31bits

I Type Instruction Set

- I type instructions format is Op code, Rs, Rt, Immediate. Here Op code is used to distinguish between instructions, Rs and Rt represents registers and Immediate field stores values.
- Total I type instructions = 16. So, minimum bits required for Op are 5 as $2^4 < 16 + 1$ reserved for R type.
- Total number of registers = 32. So, minimum bits required for Rs, Rt each are 5 bits as $2^5 = 32$.
- Immediate field stores value used in performing instructions. Upper limit of immediate can be set by setting no. of bits. But for some instructions immediate field must contain specific number of bits like for Lui (load upper immediate) which loads immediate in upper half of Rt of size 32 bits.
- If Imm. field is of size other than 16 bit then pseudo instruction Li (loading immediate value of 32 bits) will have to be changed. As it requires Lui to store 16 bits in upper half. If Imm. is of 8 bits or less than that then we have to introduce new instructions to load values in 2nd, 3rdand 4th and quarters and change Li accordingly.
- This will lead to huge loss in range of operations. As we aren't able to do operations on large values like we cannot compare 2^{15} and 2^{16}. Our operations will be restricted to very low value compared to 2^{16}. So, reducing immediate field and adding new instructions is not a good idea as we can reduce ISA but we have to restrict operations with huge offset.
- Now, considering Op code of 5 bits, Rs and Rt of 5 bits each and Imm. of 16 bits. Total number of bits for I type instruction is now 5 + 5 + 5 + 16 = 31 bits.

Instruction	**Meaning**	**Format**			
		Op^5	**Rs^5**	**Rt^5**	**Immediate^{16}**
Addi	\$s1 = \$s2 + 10	00001	rs = \$s2	rt= \$s1	imm^{16}
Addiu	\$s1 = \$s2 + 10	00010	rs = \$s2	rt= \$s1	imm^{16}
Andi	\$s1 = \$s2 & 10	00011	rs = \$s2	rt= \$s1	imm^{16}
Ori	\$s1 = \$s2 | \$s3	00100	rs = \$s2	rt= \$s1	imm^{16}
Xori	\$s1 = \$s2 ^ 10	00101	rs = \$s2	rt= \$s1	imm^{16}
Lui	\$s1 = 10 << 16	00110	0	rt= \$s1	imm^{16}
Lw	rt = MEM[rs+imm16]	00111	rs = \$s2	rt= \$s1	imm^{16}
Sw	MEM[rs+imm16] = rt	01000	rs = \$s2	rt= \$s1	imm^{16}
Beq	branch if (rs == rt)	01001	rs = \$s2	rt= \$s1	imm^{16}
Bne	branch if (rs != rt)	01010	rs = \$s2	rt= \$s1	imm^{16}
Blez	branch if (rs <= 0)	01011	rs = \$s2	0	imm^{16}
Bgtz	branch if (rs > 0)	01100	rs = \$s2	0	imm^{16}
Bltz	branch if (rs < 0)	01101	rs = \$s2	0	imm^{16}
Bgez	branch if (rs >= 0)	01110	rs = \$s2	1	imm^{16}
Slti	rt=(rs<imm?1:0)	01111	rs = \$s2	rt= \$s1	imm^{16}
Sltiu	rt=(rs<imm?1:0)	10000	rs = \$s2	rt= \$s1	imm^{16}

J Type Instruction Set

- J type instruction format is Op code, Immediate. Here Op code is used to distinguish between instructions and Immediate field stores values for changing program counter to move from one instruction to another.
- Total J type instructions : 1. So, mini. bits required for Op code is 1. Imm. field can also be reduced to lesser number of bits then 26. But it will pose restrictions on movement by Jump instruction.
- But as we know all kind of instructions of each type should be of same number of bits. Since I type can be reduced to only 31 bits so, J type should be of 31 bits. Moreover, for same Op code size for all inst. Op code should be of 5 bits So, new distribution will be Op code – 5 bits and Imm. of 26 bits.
- Total number of bits for J type instruction is = 5 + 26 = 31 bits.

Instruction	**Meaning**	**Format**	
		Op^5	Imm^{26}
J	Jump to label	10001	Imm^{26}

R Type Instruction Set

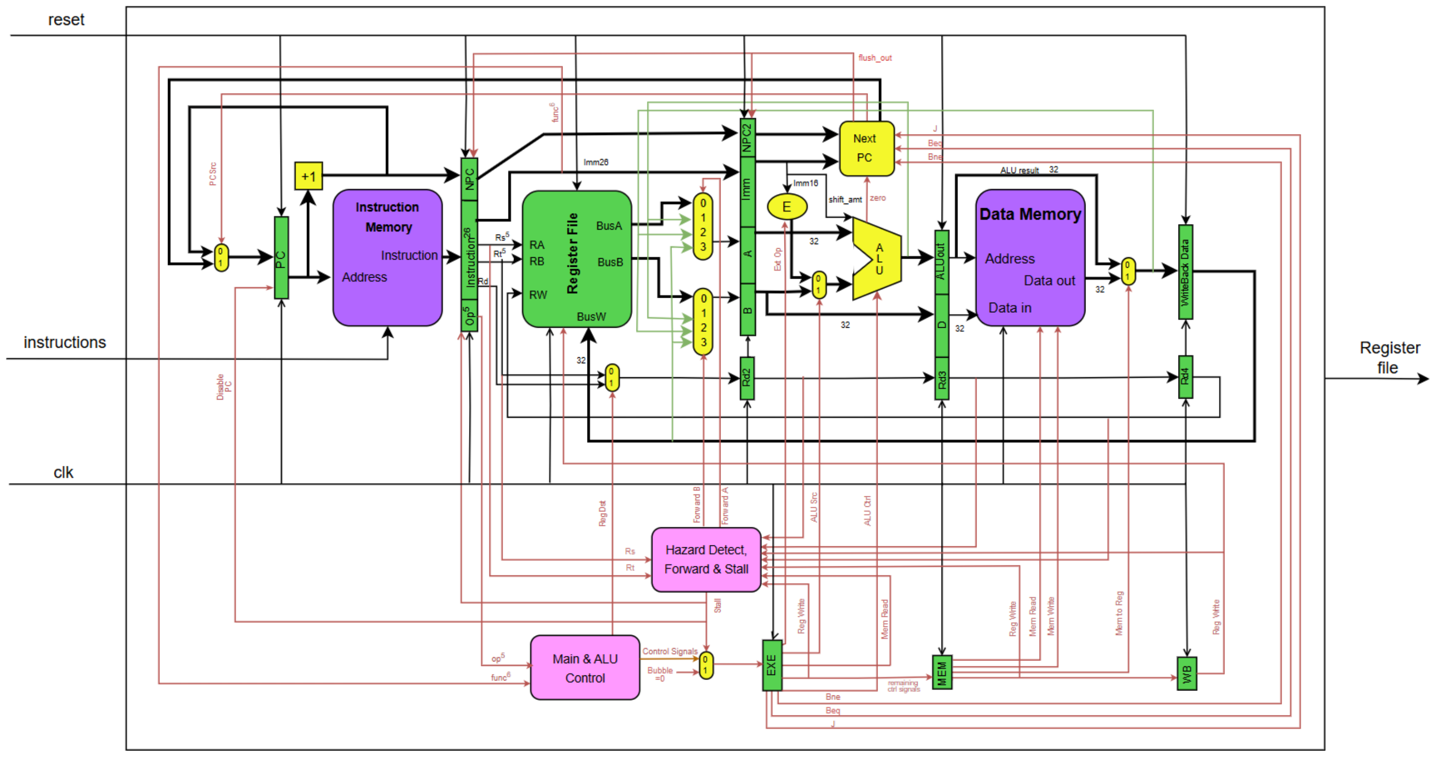
- R type instructions format is Op code, Rs, Rt, Rd, Sa, Function. Here combination of Op code and Func. bits distinguish between instructions.
- Total R type instructions = 14. So, minimum bits required for Func. is 4 bits as 2^4 = 16 and for Op code it is 0 bits as Func. alone can distinguish b/w R type instructions.
- Total number of registers = 32. So, mini. bits required for Rs, Rt, Rd and Sa each are 5 bits as 2^5 = 32.
- Sa field can be eliminated as when Sa is non-zero Rs is 0 and vice-versa. So, we can check if Func. is corresponding to Sll, Srl, Sra then consider Rs as Sa otherwise consider Rs as regular.
- So, Total number of bits required for R type instructions: 0 + 5 x 3(due to elimination of Sa) + 4= 19 bits.
- But as we know all kind of instructions of each type should be of same number of bits. Since I type can be reduced to only 31 bits so, R type should be of 31 bits. Moreover, for same Op code size for all inst. Op code should be of 5 bits So, new distribution will be Op code – 5 bits; Rs, Rt, Rd, Sa – 5 bits each and Func. of 6 bits. Total number of bits for R type instruction is = 5 + 5 x 4 + 6 = 31 bits.

Instruction	**Meaning**	**Format**					
		Op^5	**Rs^5**	**Rt^5**	**Rd^5**	**Sa^5**	**Func^6**
Add	\$s1 = \$s2 + \$s3	00000	rs = \$s2	rt= \$s3	rd = \$s1	sa = 0	000000
Addu	\$s1 = \$s2 + \$s3	00000	rs = \$s2	rt= \$s3	rd = \$s1	sa = 0	000001
Sub	\$s1 = \$s2 – \$s3	00000	rs = \$s2	rt= \$s3	rd = \$s1	sa = 0	000010
Subu	\$s1 = \$s2 – \$s3	00000	rs = \$s2	rt= \$s3	rd = \$s1	sa = 0	000011
And	\$s1 = \$s2 & \$s3	00000	rs = \$s2	rt= \$s3	rd = \$s1	sa = 0	000100
Or	\$s1 = \$s2 | \$s3	00000	rs = \$s2	rt= \$s3	rd = \$s1	sa = 0	000101
Xor	\$s1 = \$s2 ^ \$s3	00000	rs = \$s2	rt= \$s3	rd = \$s1	sa = 0	000110
Nor	\$s1 = ~(\$s2|\$s3)	00000	rs = \$s2	rt= \$s3	rd = \$s1	sa = 0	000111
Sll	\$s1 = \$s3 << 10	00000	rs = 0	rt= \$s3	rd = \$s1	sa = 10	001000
Srl	\$s1 = \$s3 >> 10	00000	rs = 0	rt= \$s3	rd = \$s1	sa = 10	001001
Sra	\$s1 = \$s3 << 10	00000	rs = 0	rt= \$s3	rd = \$s1	sa = 10	001010
Ror	\$s1 = \$s3(9 downto 0) & \$s3 >> 10	00000	rs = 0	rt= \$s3	rd = \$s1	sa = 10	001011
Slt	\$s1= (\$s2<\$s3?1:0)	00000	rs = \$s2	rt= \$s3	rd = \$s1	sa = 0	001100
Sltu	\$s1= (\$s2<\$s3?1:0)	00000	rs = \$s2	rt= \$s3	rd = \$s1	sa = 0	001101

Pseudo Instructions

Pseudo-Instructions	Op^5	Conversion to real instructions
Blez $s1, label	01011	slt $at, $zero, $s1 beq $at, $zero, label
Bgtz $s1, label	01100	slt $at, $zero, $s1 bne $at, $zero, label
Bltz $s1, label	01101	slt $at, $s1, $zero bne $at, $zero, label
Bgez $s1, label	01110	slt $at, $s1, $zero beq $at, $zero, label
move $s1, $s2	10010	addu Ss1, $s2, $zero
not $s1, $s2	10011	nor $s1, $s2, $s2
li $s1, 0xabcd	10100	ori $s1, $zero, 0xabcd
li $s1, 0xabcd1234	10101	lui $s1, 0xabcd ori $s1, $s1, 0x1234
sgt $s1, $s2, $s3	10110	slt $s1, $s3, $s2
blt $s1, $s2, label	10111	slt $at, $s1, $s2 bne $at, $zero, label

Block Diagram for Reduced Instruction Set Architecture



reset
instructions
clk
Register file
PCSrc
PC
+1
Instruction Memory
Instruction
Address
NPC
Instruction
Op
func
Imm26
Register File
RA
RB
RW
BusA
BusB
BusW
Rs
Rt
Rd
NPC2
Imm
A
B
Rd2
Imm16
E
Ext Op
shift_amt
zero
Next PC
J
Beq
Bne
flush_out
ALU
ALU Src
ALU Ctrl
ALU result
ALUout
D
Rd3
Data Memory
Address
Data out
Data in
WriteBack Data
Rd4
Disable PC
RegDst
Forward B
Forward A
Hazard Detect, Forward & Stall
Stall
Main & ALU Control
Control Signals
Bubble =0
EXE
MEM
WB
Reg Write
Mem Read
Mem Write
Mem to Reg
remaining ctrl signals
32