Computer Architecture
(Processor Design - Hardwared)

विद्याधनं सर्वधन प्रधानम्

Subhasis Bhattacharjee
Department of Computer Science and Engineering,
Indian Institute of Technology, Jammu

June 27, 2023

Outline

Processor Design

The aim of processor design

- Implement the entire SimpleRisc ISA
- Process the binary format of instructions
- Provide as much of performance as possible

Basic Approach

- Divide the processing into stages
- Design each stage separately

Data Path and Control Path

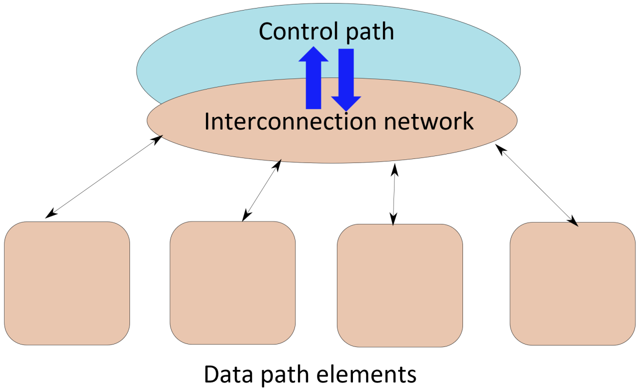
Data Path

The data path consists of all the elements in a processor that are dedicated to storing, retrieving, and processing data such as register files, memory, and the ALU.

Control Path

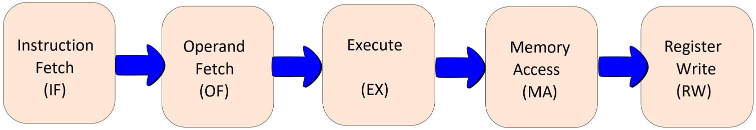
The control path primarily contains the control unit, whose role is to generate the appropriate signals to control the movement of instructions, and data in the data path.

Control Path



[We will look at the hardwired control path]

A Processor Divided Into Stages



Instruction Fetch (IF)

Instruction Fetch (IF)

- Fetch an instruction from the instruction memory
- Compute the address of the next instruction

Operand Fetch (OF) Stage

Operand Fetch (OF)

- Decode the instruction (break it into fields)
- Fetch the register operands from the register file
- Compute the branch target (PC + offset)
- Compute the immediate (16 bits + 2 modifiers)
- Generate control signals (we will see later)

Execute (EX) Stage

Execute (EX)

- The EX Stage
- Contains an Arithmetic-Logical Unit (ALU)
- This unit can perform all arithmetic operations (add, sub, mul, div, cmp, mod), and logical operations (and, or, not)
- Contains the branch unit for computing the branch condition (beq, bgt)
- Contains the flags register (updated by the cmp instruction)

MA and RW Stages

MA (Memory Access) Stage

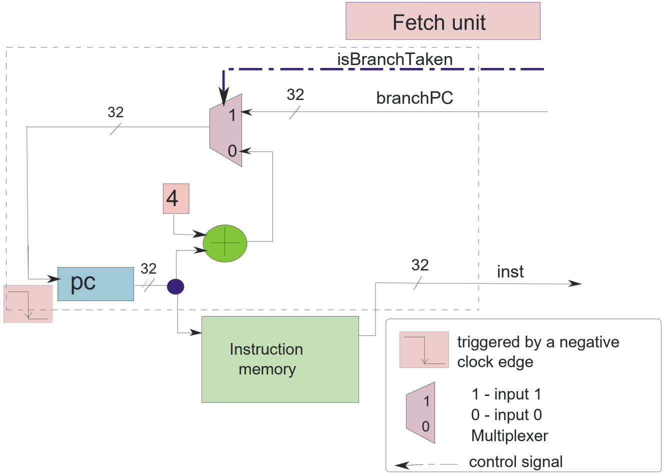
- Interfaces with the memory system
- Executes a load or a store

RW (Register Write) Stage

- Writes to the register file
- In the case of a call instruction, it writes the return address to register, ra

Instruction Fetch Unit

Instruction Fetch (IF) Stage



[negative edge triggered - assumed]

The Fetch unit

- The pc register contains the program counter
- We use the pc to access the instruction memory
- The multiplexer chooses between
  1. pc + 4
  2. branchTarget
- It uses a control signal $\rightarrow$ isBranchTaken

isBranchTaken - Signal

- isBranchTaken is a control signal
- It is generated by the EX unit (We will see it in the EX unit)

Conditions on isBranchTaken

Instruction	**Value of isBranchTaken**
non-branch instruction	0
call	1
ret	1
b	1
beq	branch taken = 1
	branch not taken = 0
bgt	branch taken = 1
	branch not taken = 0

Data Path and Control Path - revisiting

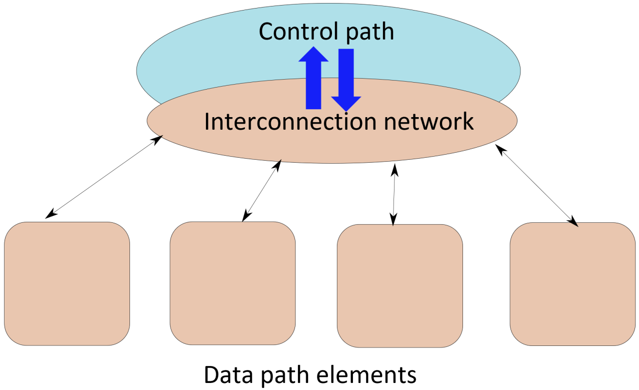
Data Path

The data path consists of all the elements in a processor that are dedicated to storing, retrieving, and processing data such as register files, memory, and the ALU.

Control Path

The control path primarily contains the control unit, whose role is to generate the appropriate signals to control the movement of instructions, and data in the data path.

Control Path

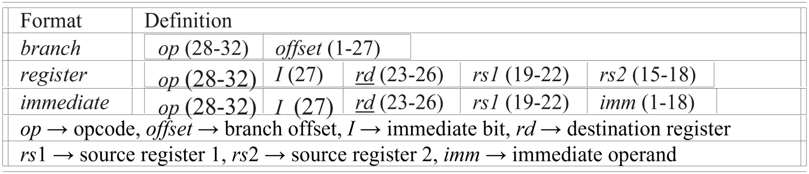


Instruction Set & Format

Instruction Set - at a glance

Inst.	Code	Format	Inst.	Code	Format
add	00000	add rd, rs1, (rs2/imm)	lsl	01010	lsl rd, rs1, (rs2/imm)
sub	00001	sub rd, rs1, (rs2/imm)	lsr	01011	lsr rd, rs1, (rs2/imm)
mul	00010	mul rd, rs1, (rs2/imm)	asr	01100	asr rd, rs1, (rs2/imm)
div	00011	div rd, rs1, (rs2/imm)	nop	01101	nop
mod	00100	mod rd, rs1, (rs2/imm)	ld	01110	ld rd, imm[rs1]
cmp	00101	cmp rs1, (rs2/imm)	st	01111	st rd, imm[rs1]
and	00110	and rd, rs1, (rs2/imm)	beq	10000	beq offset
or	00111	or rd, rs1, (rs2/imm)	bgt	10001	bgt offset
not	01000	not rd, (rs2/imm)	b	10010	b offset
mov	01001	mov rd, (rs2/imm)	call	10011	call offset
			ret	10100	ret

Instruction Formats



Format	Definition				
branch	*op* (28-32)	*offset* (1-27)			
register	*op* (28-32)	*I* (27)	<u>*rd*</u> (23-26)	*rs1* (19-22)	*rs2* (15-18)
immediate	*op* (28-32)	*I* (27)	<u>*rd*</u> (23-26)	*rs1* (19-22)	*imm* (1-18)

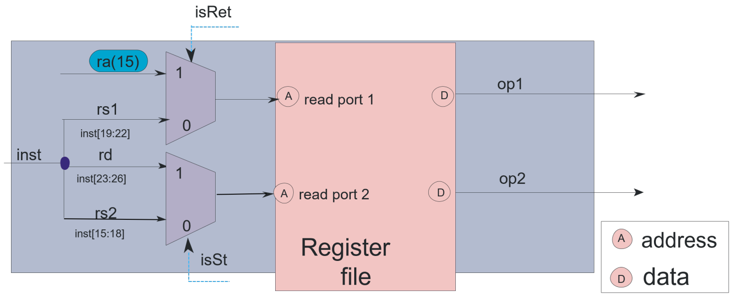
op → opcode, *offset* → branch offset, *I* → immediate bit, *rd* → destination register

*rs*1 → source register 1, *rs*2 → source register 2, *imm* → immediate operand

[Each format needs to be handled separately.]

Register File

Register File Read



Two Input Lines

- First input → rs1 or ra(15) (ret instruction)
- Second input → rs2 or rd (store instruction)

Register File Access - Input / Output

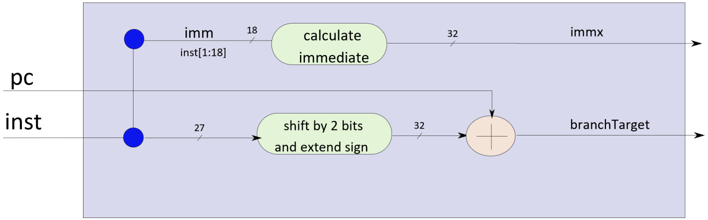
The register file has two read ports

- 1st Input
- 2nd Input

The register file has two outputs (op1, and op2)

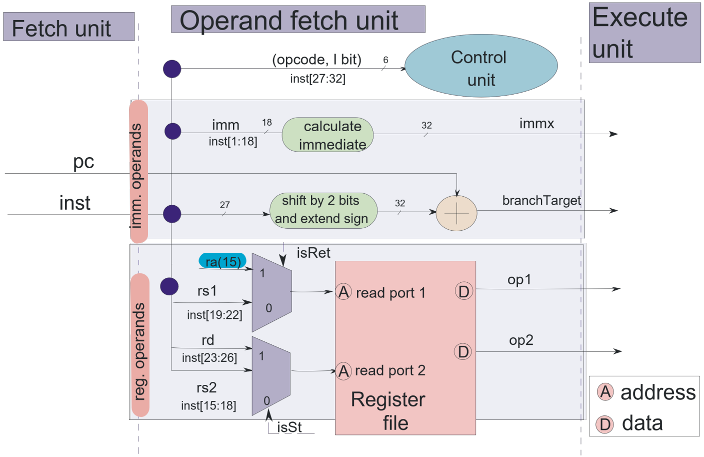
- op1 is the branch target (return address) in the case of a ret instruction, or rs1
- op2 is the value that needs to be stored in the case of a store instruction, or rs2

Immediate and Branch Unit



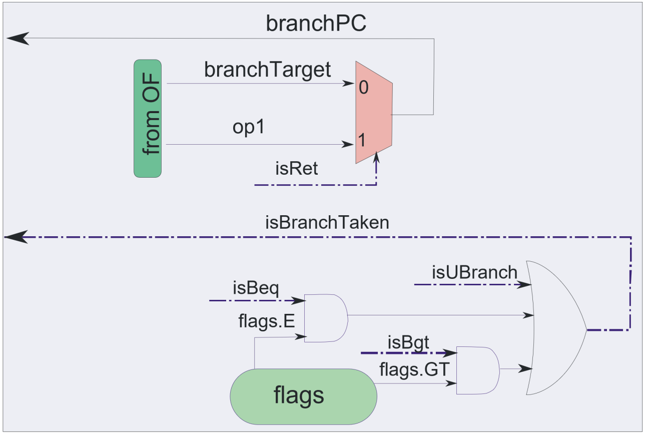
- Compute immx (extended immediate), branchTarget, irrespective of the instruction format.
- For the branchTarget we need to choose between the embedded target and op1 (ret)

OF Unit



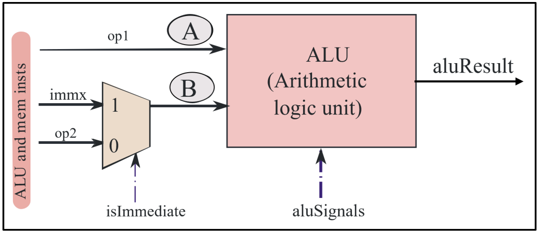
Fetch unit
Operand fetch unit
Execute unit
(opcode, I bit)
6
inst[27:32]
Control unit
imm. operands
imm
18
inst[1:18]
calculate immediate
32
immx
pc
inst
27
shift by 2 bits and extend sign
32
branchTarget
reg. operands
ra(15)
isRet
1
0
rs1
inst[19:22]
rd
inst[23:26]
1
0
rs2
inst[15:18]
isSt
A read port 1
D
A read port 2
D
Register file
op1
op2
A address
D data

EX Stage – Branch Unit



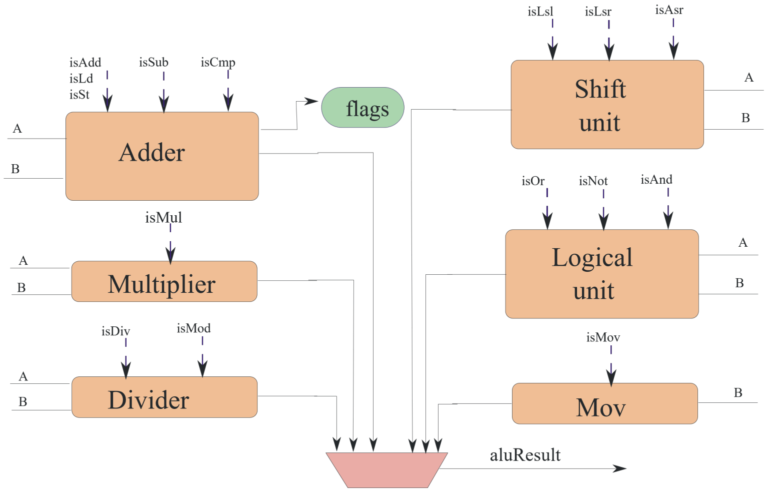
[Generates the **isBranchTaken** Signal]

ALU

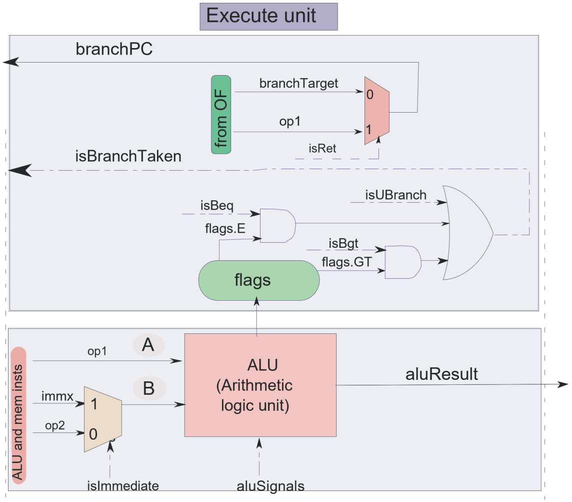


[Choose between immx and op2 based on the value of the I bit]

Inside the ALU

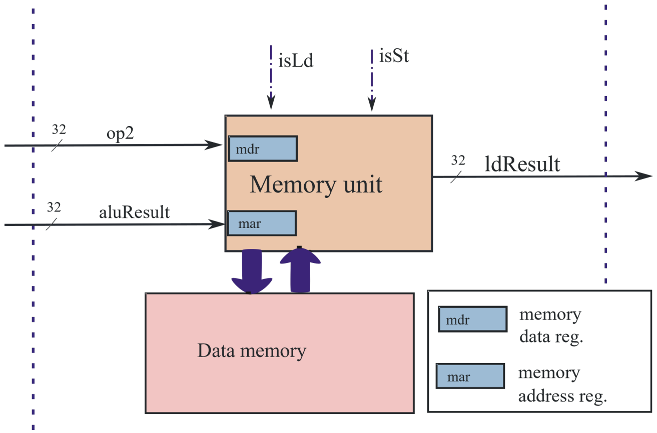


EX Unit

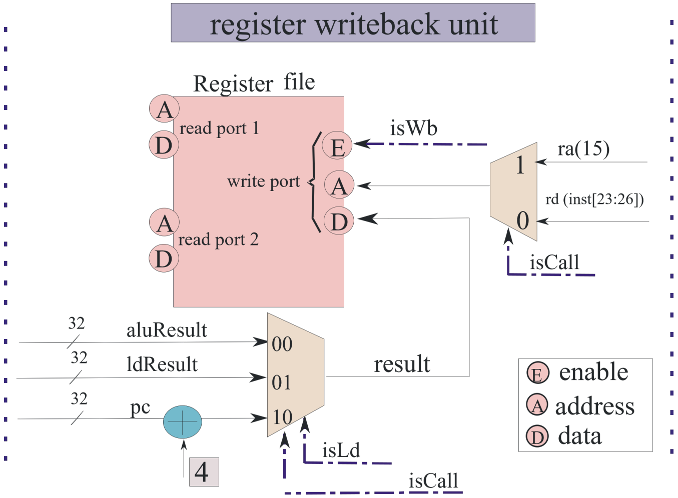


Execute unit
branchPC
from OF
branchTarget
op1
0
1
isRet
isBranchTaken
isUBranch
isBeq
flags.E
isBgt
flags.GT
flags
ALU and mem insts
op1
A
immx
B
op2
1
0
ALU
(Arithmetic
logic unit)
aluResult
isImmediate
aluSignals

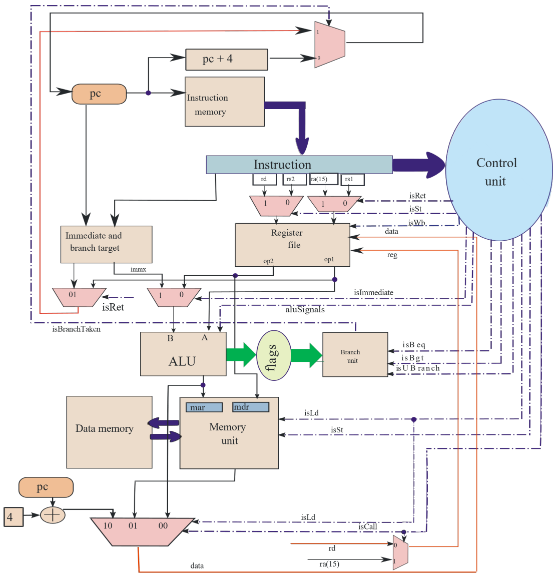
MA Unit


isLd
isSt
32
op2
mdr
Memory unit
mar
32
aluResult
32
ldResult
Data memory
mdr
memory data reg.
mar
memory address reg.

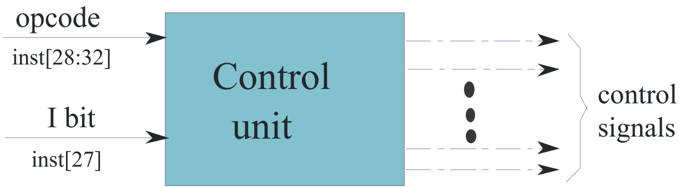
RW Unit


register writeback unit
Register file
A
read port 1
D
write port
E
A
D
A
read port 2
D
isWb
ra(15)
1
rd (inst[23:26])
0
isCall
32
aluResult
32
ldResult
32
pc
4
00
01
10
result
isLd
isCall
E enable
A address
D data

Internal Details Of Five Stage Processor



The Hardwired Control Unit



Given the opcode and the immediate (I) bit

It generates all the control signals.

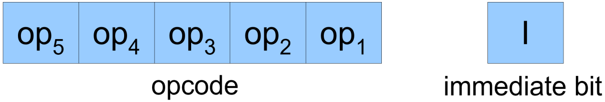
Control Signals

Sr. No.	Signal	Meaning / Explanation
1	isSt	Instruction: st
2	isLd	Instruction: ld
3	isBeq	Instruction: beq
4	isBgt	Instruction: bgt
5	isRet	Instruction: ret
6	isImmediate	I bit set to 1
7	isWb	Instructions: add, sub, mul, div, mod, and, or, not, mov, ld, lsl, lsr, asr, call
8	isUBranch	Instructions: b, call, ret
9	isCall	Instructions: call

Control Signals – II

aluSignal		
Sr. No.	Signal	Meaning / Explanation
10	isAdd	Instructions: add, ld, st
11	isSub	Instruction: sub
12	isCmp	Instruction: cmp
13	isMul	Instruction: mul
14	isDiv	Instruction: div
15	isMod	Instruction: mod
16	isLsl	Instruction: lsl
17	isLsr	Instruction: lsr
18	isAsr	Instruction: asr
19	isOr	Instruction: or
20	isAnd	Instruction: and
21	isNot	Instruction: not
22	isMov	Instruction: mov

Bits of Instructions to Generate These Signals



Control Signals

Sr. No.	Signal	Condition / Boolean Logic
1	isSt	$\overline{op5}.op4.op3.op2.op1$
2	isLd	$\overline{op5}.op4.op3.op2.\overline{op1}$
3	isBeq	$op5.\overline{op4}.\overline{op3}.\overline{op2}.\overline{op1}$
4	isBgt	$op5.\overline{op4}.\overline{op3}.\overline{op2}.op1$
5	isRet	$op5.\overline{op4}.op3.\overline{op2}.\overline{op1}$
6	isImmediate	I
7	isWb	$\neg(op5 + \overline{op5}.op3.op1.(op4 + op2)) + op5.\overline{op4}.\overline{op3}.op2.op1$
8	isUBranch	$op5.\overline{op4}.(\overline{op3}.op2 + op3.\overline{op2}.\overline{op1})$
9	isCall	$op5.\overline{op4}.\overline{op3}.op2.op1$

Control Signals – II

aluSignal		
Sr. No.	Signal	Meaning / Explanation
10	isAdd	$\overline{op5}.\overline{op4}.\overline{op3}.\overline{op2}.\overline{op1} + \overline{op5}.op4.op3.op2$
11	isSub	$\overline{op5}.\overline{op4}.\overline{op3}.\overline{op2}.op1$
12	isCmp	$\overline{op5}.\overline{op4}.op3.\overline{op2}.op1$
13	isMul	$\overline{op5}.\overline{op4}.\overline{op3}.op2.\overline{op1}$
14	isDiv	$\overline{op5}.\overline{op4}.\overline{op3}.op2.op1$
15	isMod	$\overline{op5}.\overline{op4}.op3.\overline{op2}.\overline{op1}$
16	isLsl	$\overline{op5}.op4.\overline{op3}.op2.\overline{op1}$
17	isLsr	$\overline{op5}.op4.\overline{op3}.op2.op1$
18	isAsr	$\overline{op5}.op4.op3.\overline{op2}.\overline{op1}$
19	isOr	$\overline{op5}.\overline{op4}.op3.op2.op1$
20	isAnd	$\overline{op5}.\overline{op4}.op3.op2.\overline{op1}$
21	isNot	$\overline{op5}.op4.\overline{op3}.\overline{op2}.\overline{op1}$
22	isMov	$\overline{op5}.op4.\overline{op3}.\overline{op2}.op1$