CS31001 COMPUTER ORGANIZATION AND ARCHITECTURE

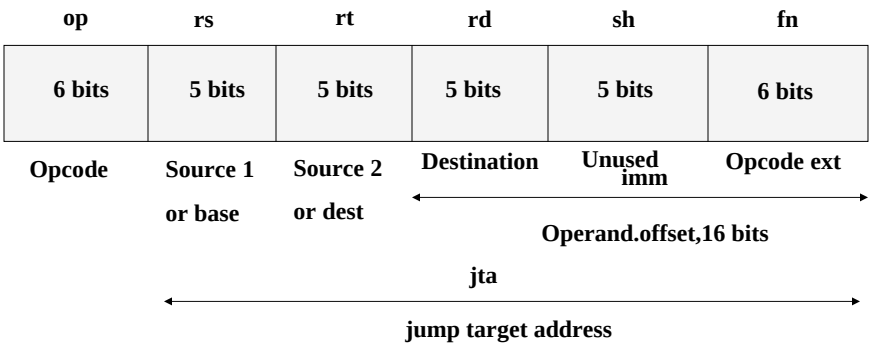
Debdeep Mukhopadhyay,
CSE, IIT Kharagpur

Instruction Execution Steps: The Single Cycle Circuit

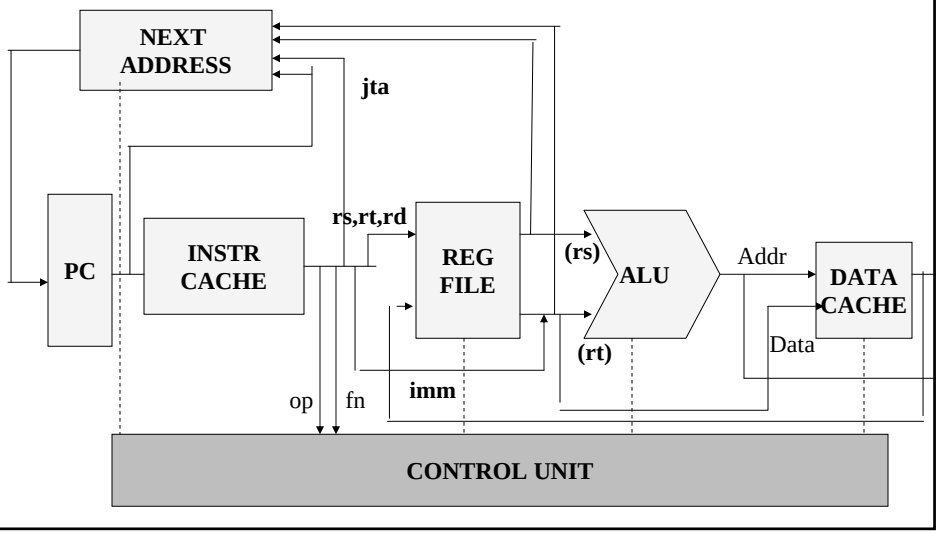
The Micro Mips ISA

Class	Instruction	Usage	Meaning	op	fn
Copy	Load upper immediate	lui rt, imm	$rt \leftarrow (imm, 0x0000)$	15	
Arithmetic	Add	add rd,rs,rt	$rd \leftarrow (rs) + (rt)$	0	32
	Subtract	sub rd,rs,rt	$rd \leftarrow (rs) - (rt)$	0	34
	Set less than	slt rd,rs,rt	$rd \leftarrow$ if $(rs) < (rd)$ then 1 else 0	0	42
	Add Immediate	addi rt,rs,imm	$rt \leftarrow (rs) + imm$	8	
	Set Less than immediate	slti rt,rs,imm	$rt \leftarrow if(rs) < imm$then 1 else 0	10	
Logic	AND	and rd,rs,rt	$rd \leftarrow (rs) \wedge (rt)$	0	36
	OR	or rd,rs,rt	$rd \leftarrow (rs) \vee (rt)$	0	37
	XOR	xor rd,rs,rt	$rd \leftarrow (rs) \oplus (rt)$	0	38
	NOR	nor rd,rs,rt	$rd \leftarrow ((rs) \vee (rt))'$	0	39
	AND immediate	andi rt,rs,imm	$rt \leftarrow (rs) \wedge imm$	12	
	OR immediate	ori rt,rs,imm	$rt \leftarrow (rs) \vee imm$	13	
	XOR immediate	xori rt,rs,imm	$rt \leftarrow (rs) \oplus imm$	14	
Memory Word	Load Word	lw rt,imm(rs)	$rt \leftarrow mem[(rs) + imm]$	35	
	Store Word	sw rt,imm,(rs)	$mem[(rs) + imm] \leftarrow (rt)$	43	
Control transfer	Jump	j L	goto L	2	
	Jump register	jr rs	goto (rs)	0	8
	Branch on less than 0	bltz rs,L	if$(rs) < 0$ then goto L	1	
	Branch on equal	beq rs,rt,L	if $(rs) = (rt)$ then goto L	4	
	Branch on not equal	bne rs,rt,L	if$(rs) \neq (rt)$ then goto L	5	
	Jump and link	jal L	goto L; $31 \leftarrow$ (PC)+4	3	
	System call	syscall	Associated with an OS system routine	0	12

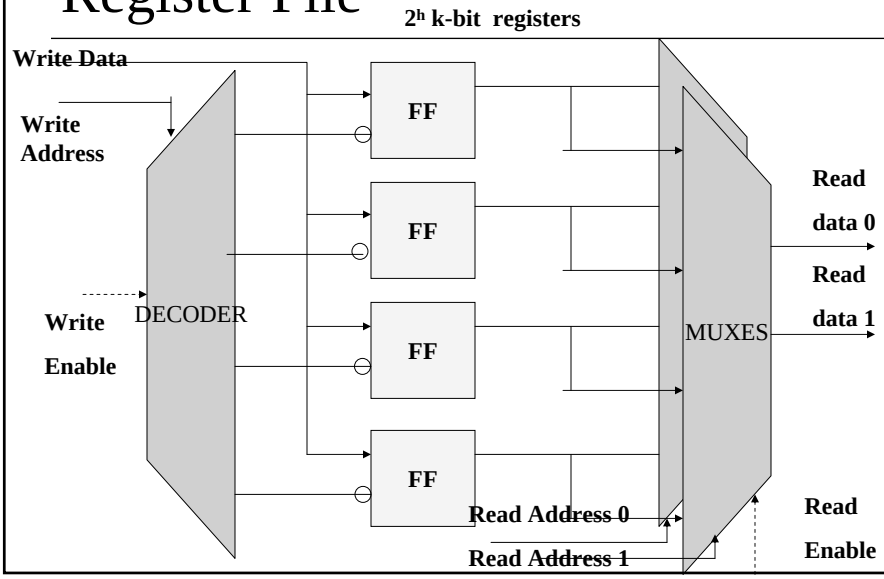
The Instruction Format



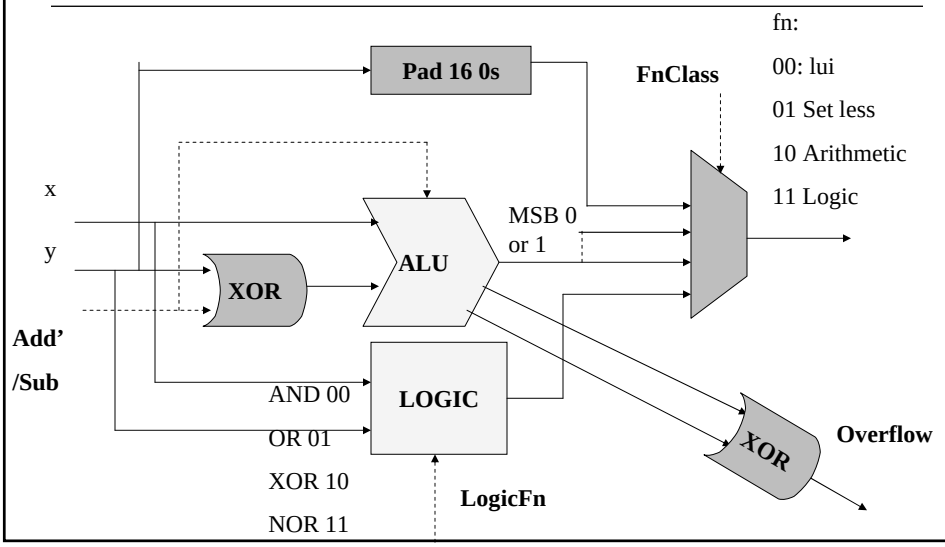
Abstraction of Instruction Execution Unit



Register File



The ALU Architecture



Execution Steps for R-type ALU Instructions

- Read contents from rs and rt, and pass them to the ALU.
- Control the ALU to perform the correct operation, according to the "func"-value of the Instruction.
- Write the output of the ALU in register rd.

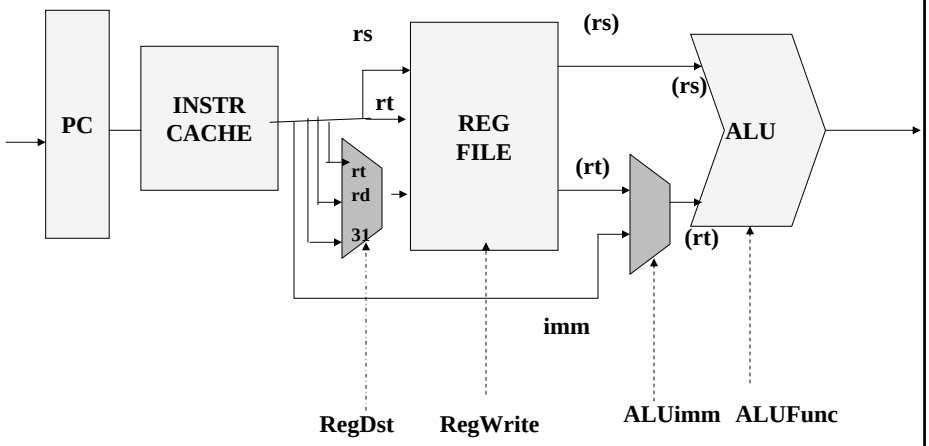
Execution Steps for I-type ALU Instructions

- Contents of rs and immediate value in the instruction are forwarded as inputs to the ALU.
- Control the ALU to perform appropriate function.
- Result is stored in the register rt (rather than rd in case of R-type instructions).

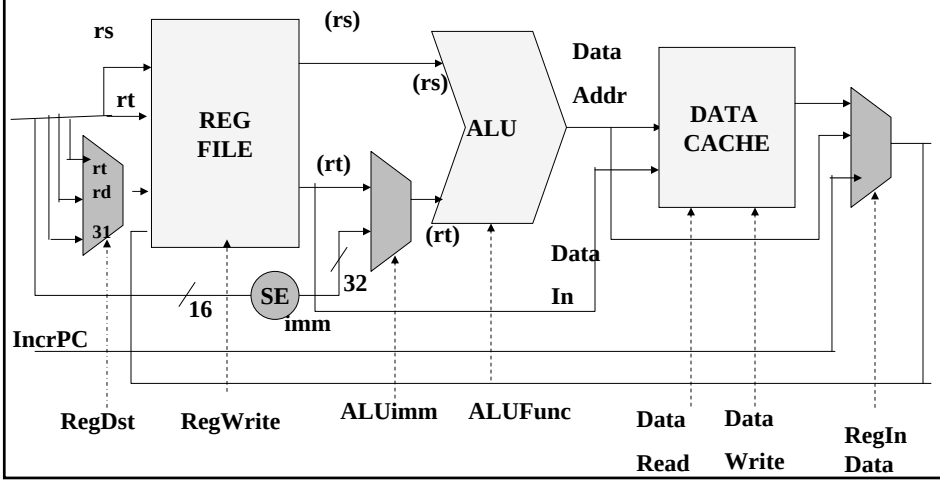
Execution Steps for I-type Memory Instructions

- Read contents of rs.
- Add the contents of rs to signed extended immediate value in the instruction to form a memory address.
- Read from or write to the memory location computed thus.
- In case of lw, place the result in rt.
- In case of sw, copy the result from rt.

Register Access and ALU



Writing of the ALU output



Handling the Branches

- The next address loaded into the program counter can be updated in various ways, depending on the type of instruction.
- Since, the addresses are always multiples of 4, the lower 2 bits are always 00.
- Hence, we consider the upper 30 bits, and consider how they can be updated.
- Thus, adding 4 to the PC value implies, that we are adding 1 to $(PC)_{31:2}$.

Update of PC in our Processor

$(PC)_{31:2}$

$=(PC)_{31:2}+1$ # Default

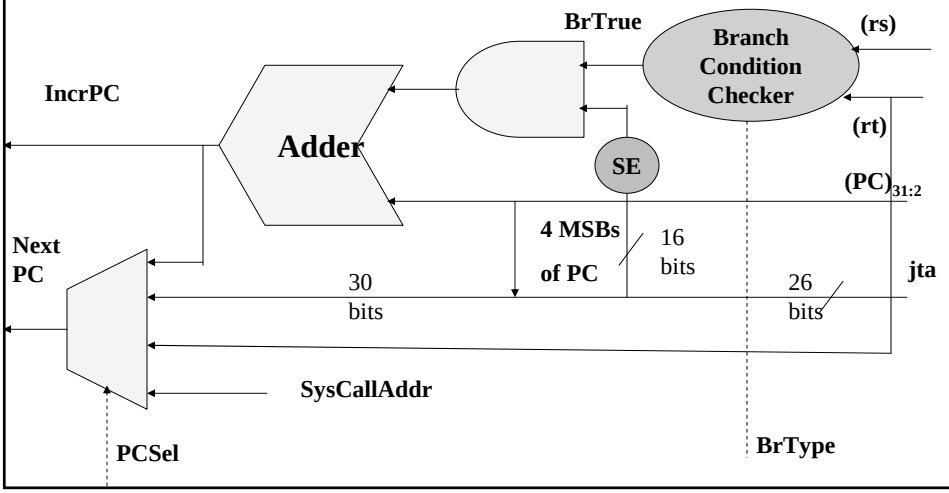
$= (PC)_{31:28}|jta$ # Unconditional jump/branch

$=(rs)_{31:2}$ #Jump Register

$= (PC)_{31:2}+1+imm$# Condition is satisfied in conditional jump

=SysCallAddr #Start Address of an OS routine

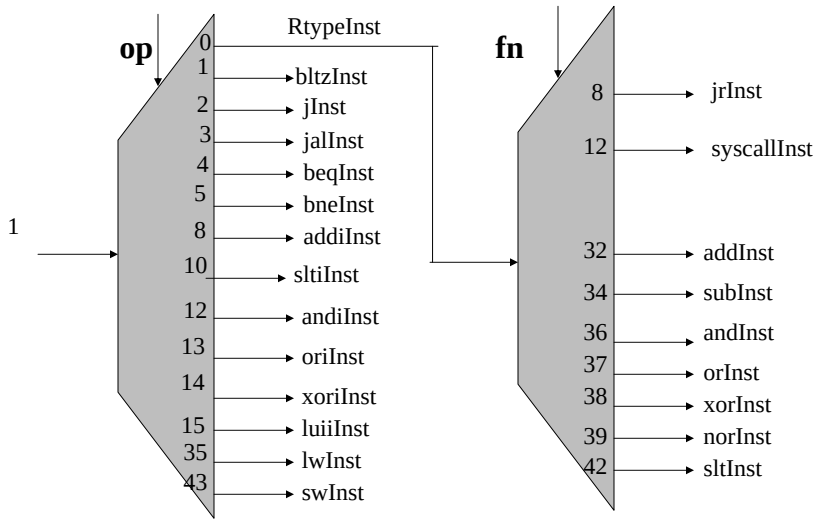
The Next Address Architecture



Generating Control Signals

- The control signals are a function of **op** and **fn** fields.
- One can express the control signals as a function of the op and fn bits.
  - It is quite easy
- However, the demerit of such an approach is if the instruction set is modified, or if we add new instructions, the entire control unit needs to be redesigned.

The Decoder based control unit



Expressing the Control Signals

- Auxiliary signals:
  - arithInst=addInst V subInst V sltInst V addiInst V sltiInst
  - logicInst=andInst V orInst V xorInst V norInst V andiInst V oriInst V xoriInst
  - immInst = luiInst V addiInst V sltiInst V andiInst V oriInst V xoriInst
- Some Control Signals:
  - RegWrite=luiInst V arithInst V logicInst V lwInst V jalInst
  - ALUImm=ImmInst V lwInst V swInst

Performance of the Single Cycle Architecture

- The above design of control circuit is a stateless and combinational design.
- Each new instruction is read from the PC, and is executed in one single clock.
  - Thus CPI=1
- The clock cycle is determined by the longest instruction.

lw is the longest instruction

- lw execution includes all the possible steps:
  1. Instruction Excess: 2 ns
  2. Register Read: 1 ns
  3. ALU operation: 2 ns
  4. Data Cache Access: 2 ns
  5. Register Write-back: 1 ns

  Total: 8 ns

  Thus a clock frequency of 125 MHz suffices.

  So, for 1 instruction, $(1/125) \times 10^{-6}$ sec

  Thus, 125 Million Instructions are executed per second (125 MIPS)

Obtaining better performance

- Note that the average instruction time is less, depends on the type of instruction, and their percentages in an application.
- Rtype 44% 6 ns No data cache

  Load 24% 8 ns

  Store 12% 7ns No register write-back

  Branch 18% 5ns Fetch+Register Read+Next-addr formation

  Jump 2% 3ns Fetch + Instruction Decode

  Weighted average = 6.36 ns

So, with a variable cycle time implementation, the performance is 157 MIPS

However, this is not possible. But we see that a single cycle implementation has a poor performance.

Summary

- Clock cycle is determined by the slowest instruction.
- If the MIPS ISA includes more complex instructions, the disadvantage is more.
  - For example if we add a MULT/DIV instruction by k times, then all operations need to be slowed down.
  - Thus MIPS does the MIPS/DIV instruction to a separate block (than the ALU block), with separate registers Hi and Lo.
    - sufficient time is kept to write back the results to the register file