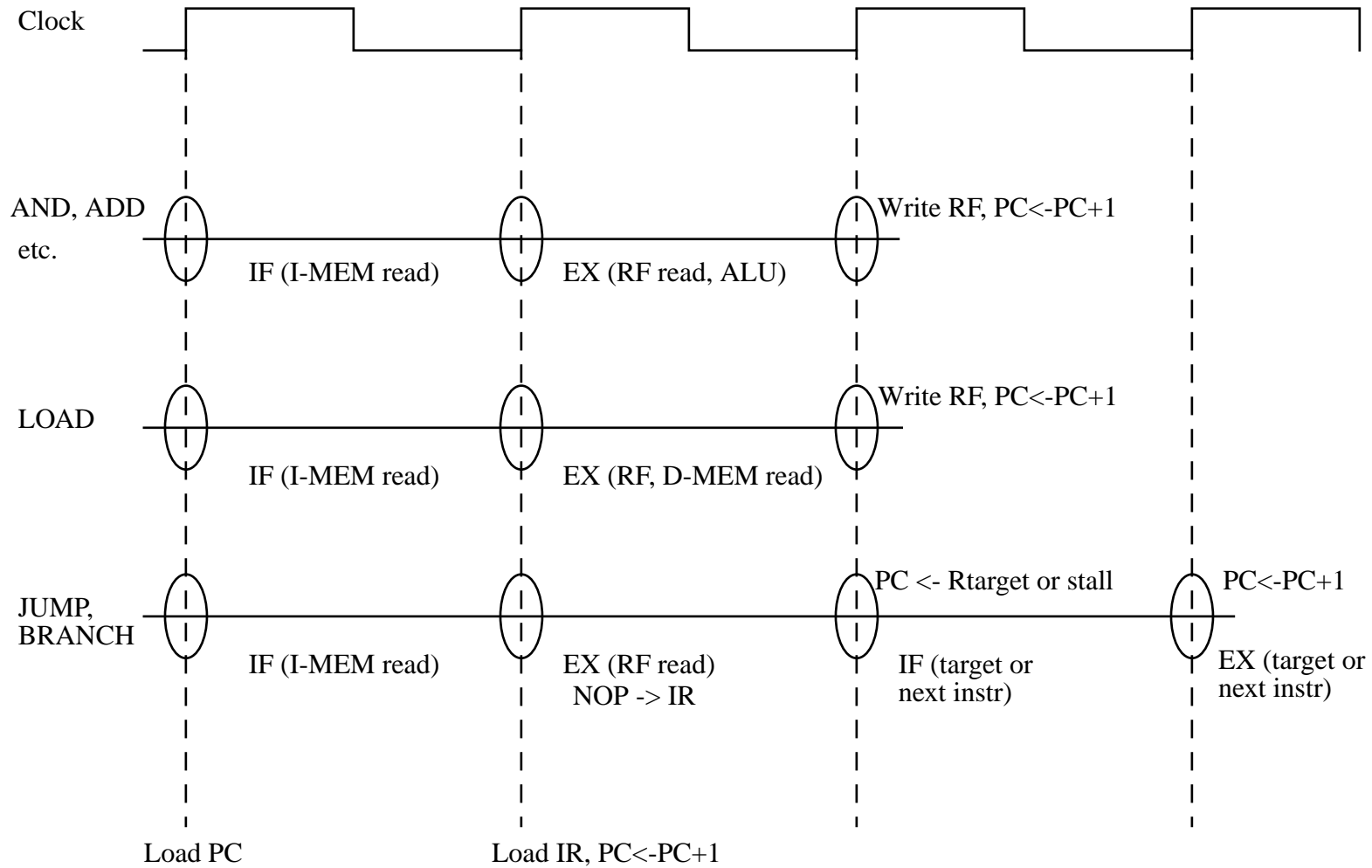


Instruction Timing for Baseline Processor



On the second clock rise the PC loads the addr of the next instruction. On the third clock rise the PC loads the addr of the next + 1 instruction for AND, ADD etc. and LOAD. It loads the target addr or remains unchanged for JP/BR. On the fourth clock rise PC points to the next + 1 instruction for untaken JP/BR, and the target + 1 for taken JP/BR.

2-Stage Pipeline for Baseline Processor Using NOPs

Code

```

or r1, r2
load r4, r3
and r3, r2
and r1, r2
    
```

Pipe Stages for load

	Fetch	Execute
clock1	or r1, r2	prev
clock2	load r4, r3	or r1, r2
clock3	and r3, r2	load r4, r3
clock4	and r1, r2	and r3, r2
clock5	next	and r1, r2

Clock Cycles for load

	clock0	clock1	clock2	clock3	clock4	clock5	clock6
prev	IF	EX					
or r1, r2		IF	EX				
load r4, r3			IF	EX			
and r3, r2				IF	EX		
and r1, r2					IF	EX	

2-Stage Pipeline for Baseline Processor Using NOPs

Code (Jump Taken)

```

or r1, r2
jmp r4
nop
and r3, r2
...
...
...
...
and r1, r2 ←
...
...

```

Pipe Stages for jmp

	Fetch	Execute
clock 1	or r1, r2	prev
clock 2	jmp r4	or r1, r2
clock 3	nop	jmp r4
clock 4	and r1, r2	nop
clock 5	next	and r1, r2

Clock Cycles for jmp

	clock 0	clock 1	clock 2	clock 3	clock 4	clock 5
prev	IF	EX				
or r1, r2		IF	EX			
jmp r4			IF	EX		
nop				IF	EX	
and r1, r2					IF	EX

2-Stage Pipeline for Baseline Processor Using NOPs

Code (Jump Not Taken)

```

or r1, r2
jmp r4
nop
and r3, r2
...
...
...
...
and r1, r2
...
...

```

Pipe Stages for jmp

	Fetch	Execute
clock 1	or r1, r2	prev
clock 2	jmp r4	or r1, r2
clock 3	nop	jmp r4
clock 4	and r3, r2	nop

Clock Cycles for jmp

	clock 0	clock 1	clock 2	clock 3	clock 4	clock 5
prev	IF	EX				
or r1, r2		IF	EX			
jmp r4			IF	EX		
nop				IF	EX	
and r3, r2					IF	EX

2-Stage Pipeline for Baseline Processor Using Stalls

Code (Jump Taken)

```

or r1, r2
jmp r4
and r3, r2
...
...
...
and r1, r2
...
...

```

Pipe Stages for jmp

	Fetch	Execute
clock 1	or r1, r2	prev
clock 2	jmp r4	or r1, r2
clock 3	and r3, r2	jmp r4
clock 4	and r1, r2	forced nop

Clock Cycles for jmp

	clock0	clock1	clock2	clock3	clock4	clock5
prev	IF	EX				
or r1, r2		IF	EX			
jmp r4			IF	EX	(New PC)	
and r3, r2				IF	stall	
and r1, r2					IF	EX

2-Stage Pipeline for Baseline Processor Using Stalls

Code (Jump Not Taken)

```

or r1, r2
jmp r4
and r3, r2
...
...
...
...
and r1, r2
    
```

Pipe Stages for jmp

	Fetch	Execute
clock 1	or r1, r2	prev
clock 2	jmp r4	or r1, r2
clock 3	and r3, r2	jmp r4
clock 4	and r3, r2	forced nop

Clock Cycles for jmp

	clock 0	clock 1	clock 2	clock 3	clock 4	clock 5
prev	IF	EX				
or r1, r2		IF	EX			
jmp r4			IF	EX		
and r3, r2				IF	stall	EX
next instr						IF

2-Stage Pipeline for Baseline Processor Using Stalls

Optimization for Untaken Jumps and Branches

- Do not stall when a jump or branch is not taken.
- Since the next instruction has already been fetched, it can be executed on the next clock cycle.
- Simple to implement on a 2-stage pipe, more complex in deeper pipes.