Single vs. Multi-cycle MIPS

Single Clock Cycle Length

- Suppose we have
  - Fetch 2ns
  - Decode 2ns
  - Register read 2ns
  - Register write 2ns
  - Memory read 2ns
  - Memory write 2ns
  - ALU 2ns
- What is the clock cycle length?

Single Cycle Length

- **Worst case propagation delay involves Load**
  - Fetch, Decode/Read regs, ALU, Read mem, Write regs
- Thus, cycle length is:
  - 2ns + 2ns + 2ns + 2ns + 2ns = 10ns
  - Decode/register read are done simultaneously
  - Clock cycle rate is 1s / 10ns = 100 MHz
- Single cycle design: ***ALL*** instruction types take 10ns!

Single Cycle Length

One clock cycle for all steps

Load	Fetch	Decode Read Regs	ALU Eff. Addr.	Mem Read	Write Reg
Arithmetic	Fetch	Decode Read Regs	ALU	Write Reg	
Branch, Jump	Fetch	Decode Read Regs	ALU Comp.		
Store	Fetch	Decode Read Regs	ALU Eff. Addr.	Mem Write	

Single Cycle Length

- How long does an **Add** take?
  - 10ns – it's a single cycle implementation
  - But notice add doesn't use the memory
  - ***It could be done in 8ns*** (fetch, decode/read registers, ALU, write registers)
- How about a Branch? (done in 6 ns but takes 10 ns)
- How about a Jump? (done in 6 ns but takes 10 ns)
- How about a Store? (done in 8 ns but takes 10 ns)

Can we do better?

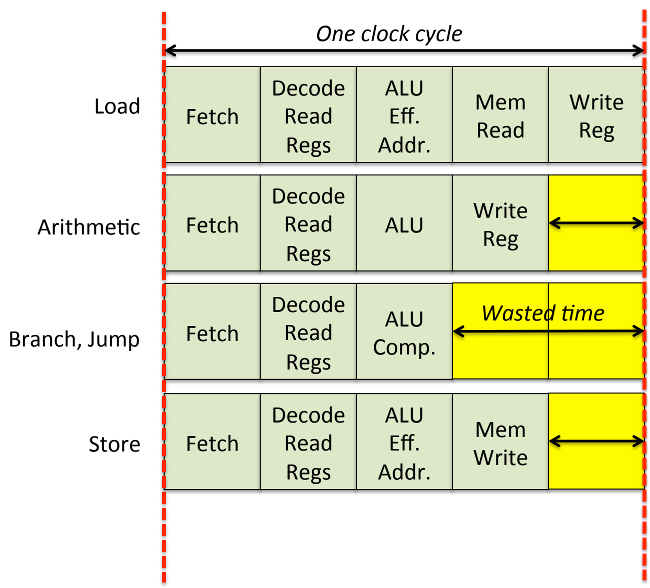
One clock cycle

Load	Fetch	Decode Read Regs	ALU Eff. Addr.	Mem Read	Write Reg	10 ns
Arithmetic	Fetch	Decode Read Regs	ALU	Write Reg		8 ns but 10 ns cycle
Branch, Jump	Fetch	Decode Read Regs	ALU Comp.			6 ns but 10 ns cycle
Store	Fetch	Decode Read Regs	ALU Eff. Addr.	Mem Write		8 ns but 10 ns cycle

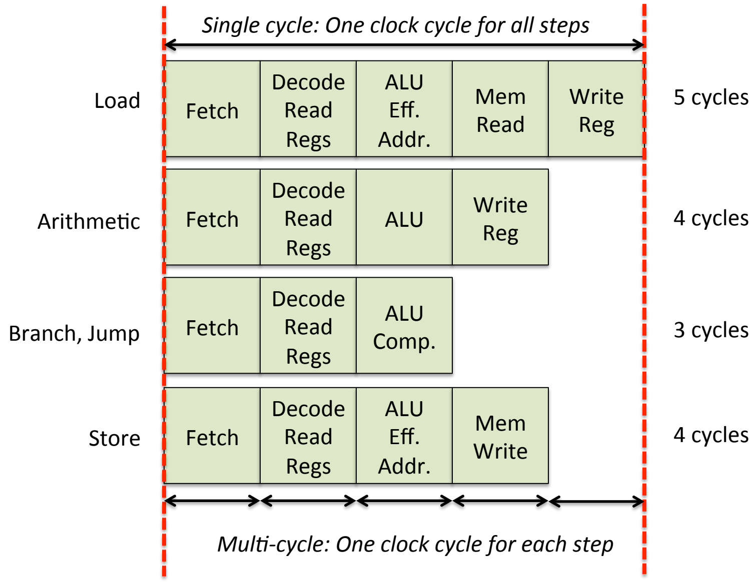
Can we do better?

- Multi-cycle implementation
  - ***Divide steps*** into their own ***shorter (faster) clock cycles***
    - Fetch: 1 cycle
    - Decode/read registers: 1 cycle
    - ALU: 1 cycle
    - Memory read/write: 1 cycle
    - Write registers: 1 cycle
- Load takes 5 cycles, add takes 4 cycles, branch takes 3 cycles (comparison done in 3rd cycle), jump takes 3 cycles and store takes 4 cycles

Can we do better?



Can we do better?



Can we do better?

- What is the clock cycle length for multi-cycle case?
  - Maximum delay of any one of the steps
    - Clock cycle length = max(time of each step)
  - In this example, the clock cycle length is 2 ns
- How long does ***each instruction type*** take now?
  - Load: 5 cycles * 2 ns/cycle = 10 ns
  - Add/r-type: 4 cycles * 2 ns/cycle = 8 ns
  - Jump, branch: 3 cycles * 2 ns/cycle = 6 ns
  - Store: 4 cycles * 2 ns/cycle = 8 ns

How does this help?

- Consider this program:

```
.data
A:      .word 10,20,30,40,50,60,70,80,90
B:      .word 0,0,0,0,0,0,0,0,0,0
.text
        li    $t0,10            # 1 instruction
        la    $t1,A             # 2 instructions
loop:   lw    $t3,0($t1)        # executed 10 times, 10 loads total
        add   $t3,$t3,$t3
        add   $t3,$t3,$t3
        sw    $t3,40($t1)       # executed 10 times, 10 stores
        addi  $t1,$t1,4
        addi  $t0,$t0,-1        # 4 adds per iteration * 10 = 40 adds
        bne   $t0,$0,loop       # executed 10 times, 10 branches
        li    $v0,10            # 1 instruction
        syscall                 # 1 instruction
```

How does this help?

- For previous program, we have the counts:
  - 45 add instructions
  - 10 load instructions
  - 10 store instructions
  - 10 branch instructions
  - Total instruction count (IC) = 75 instructions
- Suppose single cycle implementation is 10 ns cycle
- CPU time is how long program executes
- Thus, single cycle CPU time is 75 instr * 10 ns = 750ns

How does this help?

- CPU time for multi-cycle? I.e., how much time does it take to execute this program on multi-cycle.
- Each instruction type takes different number cycles
- Thus, we have in this example:

CPU time = 10 loads * 5 cycles * 2 ns/cycle +
45 adds * 4 cycles * 2 ns/cycle +
10 stores * 4 cycles * 2 ns/cycle +
10 branches * 3 cycles * 2 ns/cycle
= 600 ns

Multi-cycle is FASTER than single cycle (600 ns vs. 750 ns)

How does this help?

- Consider ratio of single cycle and multi cycle CPU times:
  - 750 ns / 600 ns = 1.25 times faster
  - The multi-cycle is 1.25 times faster than single cycle
- Speedup = Slower CPU time / Faster CPU time

Consider two programs A, B

A, B executed on single and multi-cycle MIPS impl.

A: 800 adds, 200 branches

CPU time single cycle = (800+200) × 1 cycle per instruction × 10ns = 10,000ns
CPU time multi cycle = 800 adds × 4 cycles × 2ns +
200 branches × 3 cycles × 2 ns
= 7,600ns
Speedup = 10,000 ns / 7,600 ns = 1.32x

B: 100 adds, 800 loads, 100 branches

CPU time single cycle = (100+800+100) × 1 cycle × 10ns = 10,000ns
CPU time multi cycle = 100 adds×4 cycles×2ns + 800 loads×5 cycles×2 ns +
100 branches×3 cycles×2 ns
= 9,400 ns
Speedup = 10,000 ns / 9,400 ns = 1.06x

Instruction Mix

- Speedups are vastly different in A,B due to the different instructions executed
- Instruction mix: The percentage of total instruction count (IC) corresponding to each instruction type

- A: 80% arithmetic (add), 20% branches
- B: 10% arithmetic, 80% loads, 10% branches