

Consider a 2-way set-associative cache that uses 10-bit cache index and has 32-byte cache blocks. If a machine uses 32 bit physical addresses, compute:

- the number of blocks in the cache: $2 \times 2^{10} = 2048$
- the size of the block offset: 5 Bits
- the size of the tag: 17 Bits

Tag	Index	Offset (range)
17 bits	10 bits (Given)	5 bits (#Of bits to make 32)

Consider a 2-way set-associative cache with a total of 4 blocks of 4 bytes each:

Set	Block	Block
0		
1		

Tag	Index	Offset (range)
Remaining bits	1 bit (# bits to make 0d2)	2 bits (#Of bits to make 0d4)

```

loop:  add    $a0, $a0, 4
      lw     $t0, -4($a0)
      bne    $t0, $a1, skip
      add    $v0, $v0, 1
skip:  bne    $a0, $t1, loop
  
```

Given a direct-mapped cache with 2 blocks of 2 bytes each.

Set	Block
0	
1	

Direct mapping means 1Block/set

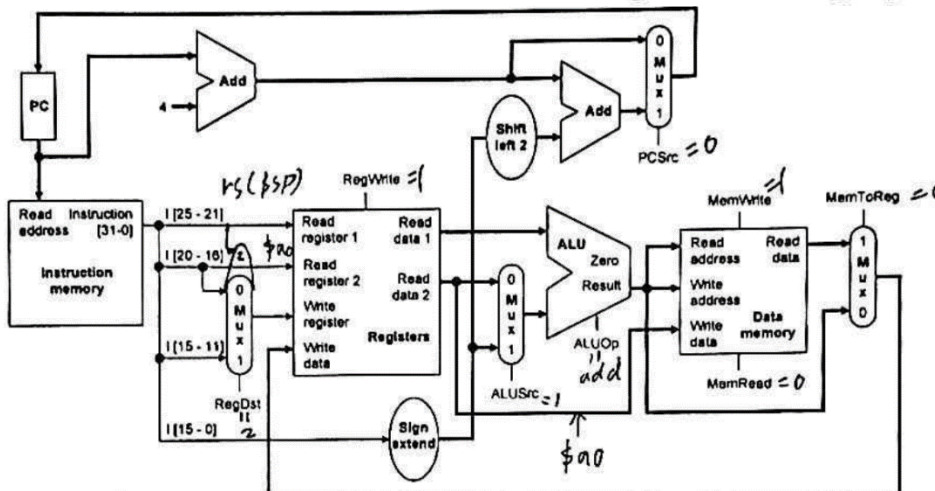
Tag	Index	Offset (range)
Remaining bits	1 bit (# bits to make 0d2)	1 bit (#Of bits to make 0d2)

Inst	iter	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
add	N	F	D	E	M	W																	
lw	N		F	D	E	M	W																
bne	N			F	D	E	M	W															
bne	N							F	D	E	M	W											
add	N+1									F	D	E	M	W									
lw	N+1									F	D	E	M	W									
bne	N+1									F	D	E	M	W									

Consider the above single-cycle datapath. We want to implement a new I-type MIPS instruction **push \$rt** which grows the stack by 4 bytes and stores \$rt onto the stack.

Example: The instruction **push \$a0** is equivalent to the MIPS instruction sequence: **addi \$sp, \$sp, -4; sw \$a0, 0(\$sp)**

Question: Make the fewest possible changes to the given datapath to implement the push instruction. Be sure to indicate the value of all control signals, including any new control signals.



Amdahl's Law

- Amdahl's Law states that optimizations are limited in their effectiveness

$$\text{Execution time after improvement} = \frac{\text{Time affected by improvement}}{\text{Amount of improvement}} + \text{Time unaffected by improvement}$$

- Example:** Suppose we double the speed of floating-point operations
- If only 10% of the program execution time T involves floating-point code, then the overall performance improves by just 5%

$$\text{Execution time after improvement} = \frac{0.10 T}{2} + 0.90 T = 0.95 T$$

