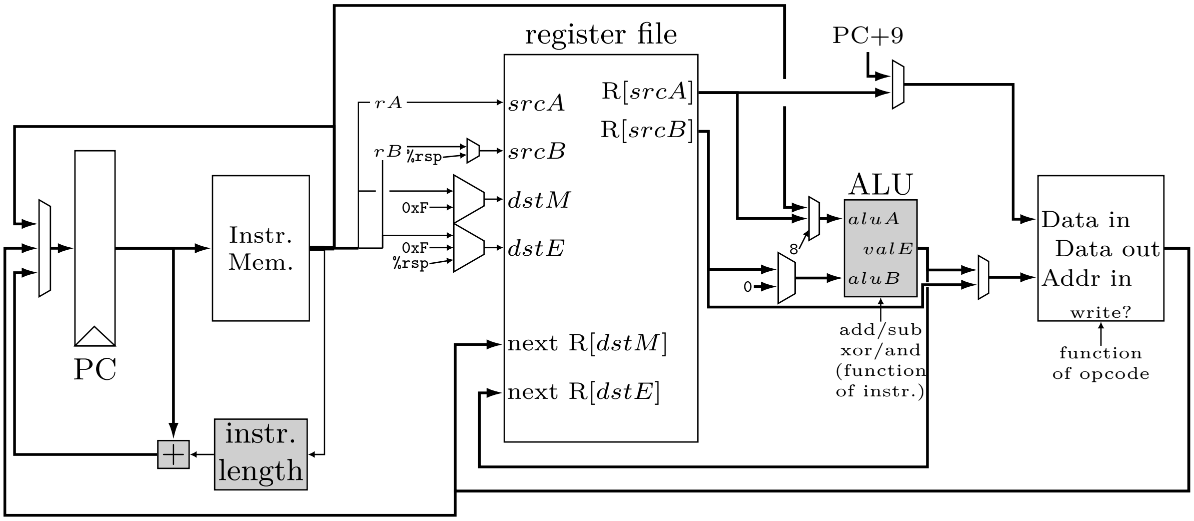
1 Y86-64 ISA

byte:	0	1	2	3	4	5	6	7	8	9
`halt`	0 0									
`nop`	1 0									
`rrmovq/cmovCC` *rA, rB*	2 *cc*	*rA* *rB*								
`irmovq` *V, rB*	3 0	F *rB*	*V*							
`rmmovq` *rA, D(rB)*	4 0	*rA* *rB*	*D*							
`mrmovq` *D(rB), rA*	5 0	*rA* *rB*	*D*							
OP`q` *rA, rB*	6 *fn*	*rA* *rB*								
`j`*CC Dest*	7 *cc*	*Dest*								
`call` *Dest*	8 0	*Dest*								
`ret`	9 0									
`pushq` *rA*	A 0	*rA* F								
`popq` *rA*	B 0	*rA* F								

2 Y86-64 Processor (single-cycle design)



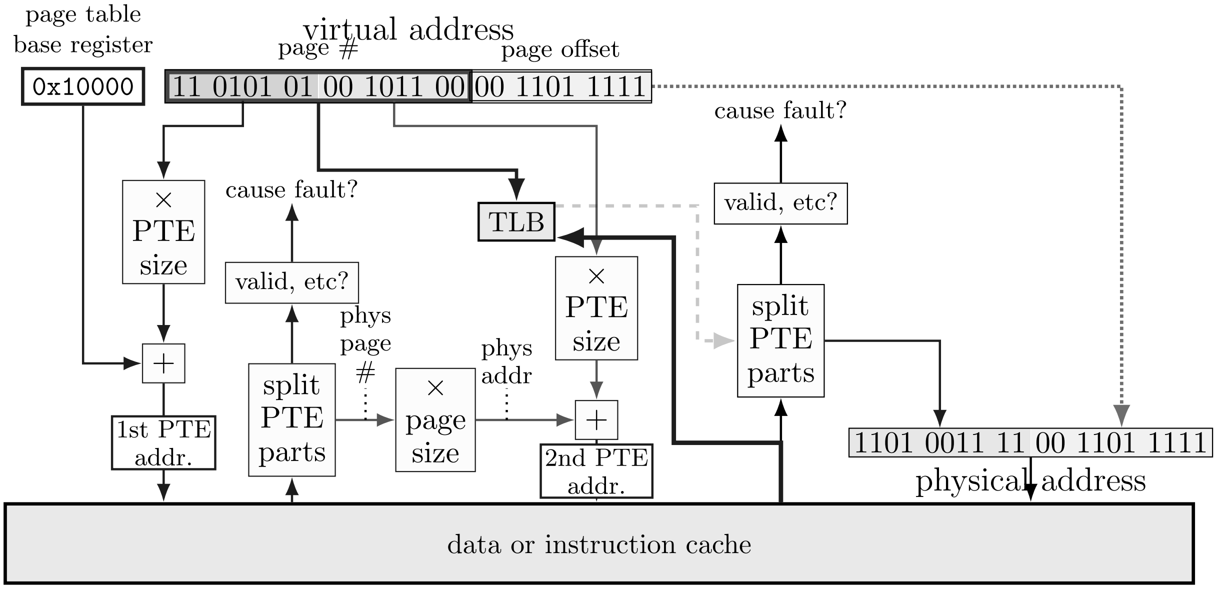
3 Linux x86-64 calling convention

- arguments (in order): `%rdi`, `%rsi`, `%rdx`, `%rcx`, `%r8`, `%r9`
- return value: `%rax`

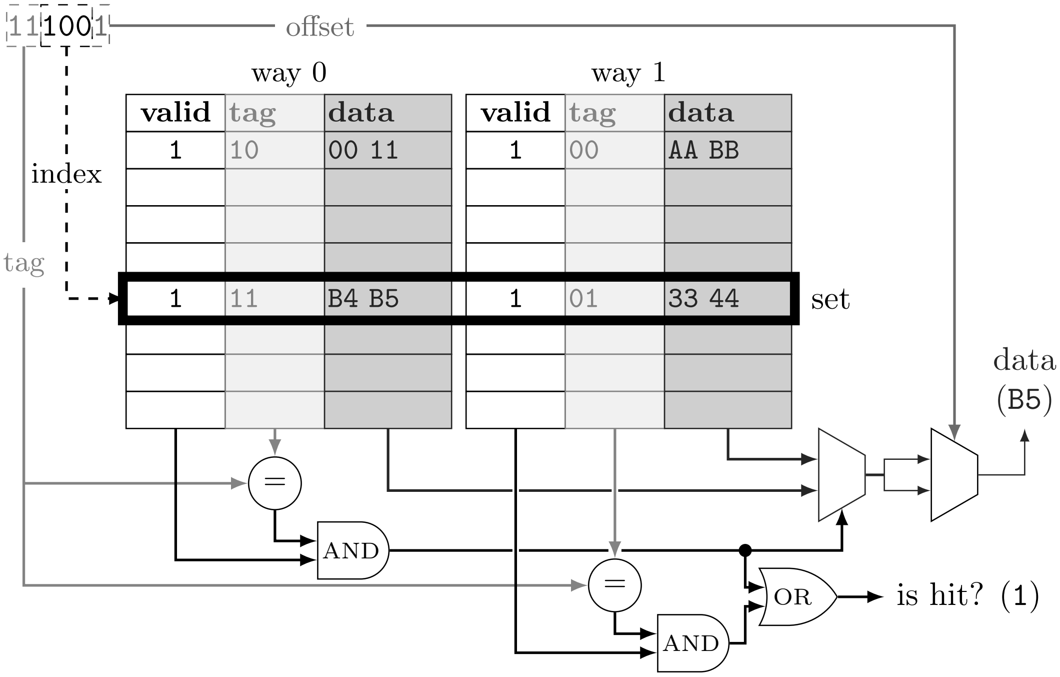
4 x86-64 assembly misc

- in AT&T syntax: `OP src, dst`; registers = %, constants = $
- `pushq X`: subtract 8 from `%rsp`; move X to memory at %rsp
- `popq X`: mov X to memory at `%rsp`; add 8 to `%rsp`
- `call X`: push return address on stack, jump to X
- `ret`: pop return address from stack, jump to it
- `0x1234(%rax, %rbx, 4)` indicates the value in memory at address `%rax + %rbx` × `4 + 0x1234`

5 page table lookup



6 cache organization



valid	tag	data	valid	tag	data
1	10	00 11	1	00	AA BB
1	11	B4 B5	1	01	33 44