virtual memory 3

last time

copy-on-write: share read-only + copy on demand

- way to efficiently (kinda) implement fork

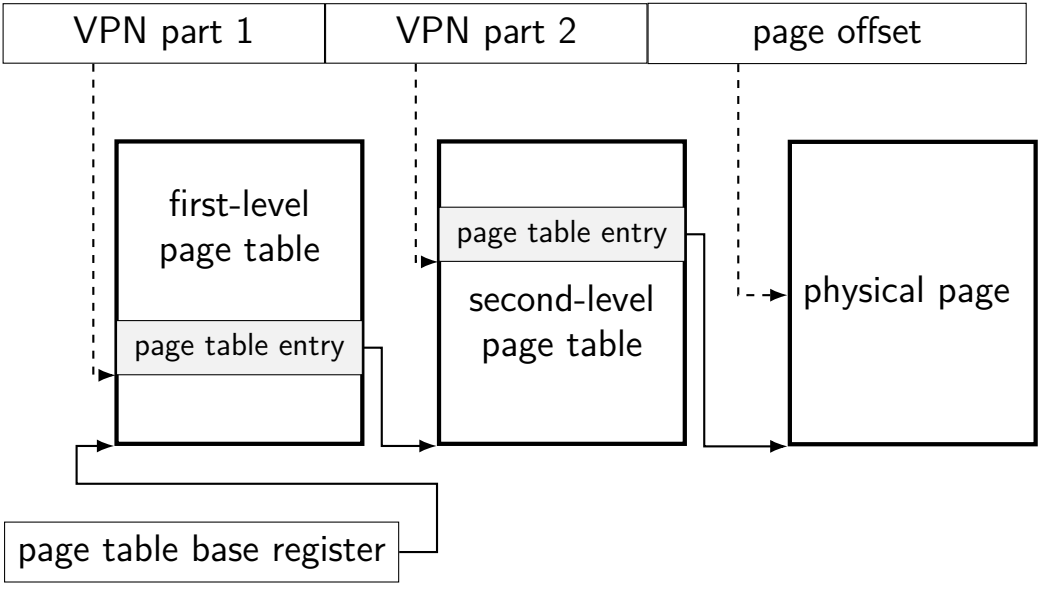
swapping: memory as a cache for disk/SSD

- hits: processor does page table lookup
- misses: OS handles, including choosing what to evict

multi-level tables

- divide virtual page number into parts
- use first part as index into first table
- first table entry locates second table
- use second part as index into second table
- ...last part entry identifies page w/ actual data

another view



2-level exercise (1)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

page table base register `0x08`; translate virtual address `0x0FB`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

2-level exercise (1)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

page table base register `0x08`; translate virtual address `0x0FB`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x0F3` = `011 111 011`

(PTE 1 addr: `0x08` + PTE size times 011 (3))

PTE 1: `0xBB` at `0x0B`

PTE 1: PPN `101` (5) valid `1`

PTE 2: `0xF0` at `0x2F`

PTE 2: PPN `111` (7) valid `1`

`111 011` = `0x3B` → `0x0C`

2-level exercise (1)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

page table base register `0x08`; translate virtual address `0x0FB`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x0F3` = `011 111 011`
(PTE 1 addr: `0x08` + PTE size times 011 (3))
PTE 1: `0xBB` at `0x0B`
PTE 1: PPN `101` (5) valid `1`
PTE 2: `0xF0` at `0x2F`
PTE 2: PPN `111` (7) valid `1`
`111 011` = `0x3B` → `0x0C`

2-level exercise (1)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

page table base register 0x08; translate virtual address 0x0FB

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

0x0F3 = 011 111 011
(PTE 1 addr: 0x08 +
PTE size times 011 (3))
PTE 1: 0xBB at 0x0B
PTE 1: PPN 101 (5) valid 1
PTE 2: 0xF0 at 0x2F
PTE 2: PPN 111 (7) valid 1
111 011 = 0x3B → 0x0C

2-level exercise (1)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

page table base register `0x08`; translate virtual address `0x0FB`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x0F3` = `011 111 011`
(PTE 1 addr: `0x08` + PTE size times 011 (3))
PTE 1: `0xBB` at `0x0B`
PTE 1: PPN `101` (5) valid `1`
PTE 2: `0xF0` at `0x2F`
PTE 2: PPN `111` (7) valid `1`
`111 011` = `0x3B` → `0x0C`

2-level exercise (2)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

page table base register `0x10`; translate virtual address `0x0F9`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 5A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

2-level exercise (2)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

page table base register `0x10`; translate virtual address `0x0F9`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 5A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x0FF` = `011 111 001`

(PTE 1 at:

0x10 + PTE size times 3 (011))

PTE 1: `0x3A` at `0x13`

PTE 1: PPN `001` (1) valid `1`

(second table at:

1 (001) times page size = 0x08)

PTE 2: `0xFF` at `0x0F`

PTE 2: PPN `111` (7) valid `1`

`111 001` = `0x39` → `0x0C`

2-level exercise (2)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

page table base register `0x10`; translate virtual address `0x0F9`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 5A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x0FF` = `011 111 001`

(PTE 1 at:

0x10 + PTE size times 3 (011))

PTE 1: `0x3A` at `0x13`

PTE 1: PPN `001` (1) valid `1`

(second table at:

1 (001) times page size = 0x08)

PTE 2: `0xFF` at `0x0F`

PTE 2: PPN `111` (7) valid `1`

`111 001` = `0x39` → `0x0C`

2-level exercise (2)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

page table base register `0x10`; translate virtual address `0x0F9`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 5A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x0FF` = `011 111 001`

(PTE 1 at:

0x10 + PTE size times 3 (011))

PTE 1: `0x3A` at `0x13`

PTE 1: PPN `001` (1) valid `1`

(second table at:

1 (001) times page size = 0x08)

PTE 2: `0xFF` at `0x0F`

PTE 2: PPN `111` (7) valid `1`

`111 001` = `0x39` → `0x0C`

2-level exercise (2)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

page table base register `0x10`; translate virtual address `0x0F9`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 5A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x0FF` = `011 111 001`

(PTE 1 at:

0x10 + PTE size times 3 (011))

PTE 1: `0x3A` at `0x13`

PTE 1: PPN `001` (1) valid `1`

(second table at:

1 (001) times page size = 0x08)

PTE 2: `0xFF` at `0x0F`

PTE 2: PPN `111` (7) valid `1`

`111 001` = `0x39` → `0x0C`

2-level exercise (3)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x08`; translate virtual address `0x00B`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

2-level exercise (3)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x08`; translate virtual address `0x00B`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x0F3` = `000 001 011`

PTE 1: `0x88` at `0x08`

PTE 1: PPN `100` (5) valid `0`

page fault!

2-level exercise (3)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x08`; translate virtual address `0x00B`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x0F3` = `000 001 011`

PTE 1: `0x88` at `0x08`

PTE 1: PPN `100` (5) valid `0`

page fault!

2-level exercise (4)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x08`; translate virtual address `0x1CB`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

2-level exercise (4)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x08`; translate virtual address `0x1CB`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x1CB` = `111 001 011`

PTE 1: `0xFF` at `0x0F`

PTE 1: PPN `111` (7) valid `1`

PTE 2: `0x0C` at `0x39`

PTE 2: PPN `000` (0) valid `0`

page fault!

2-level exercise (4)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x08`; translate virtual address `0x1CB`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x1CB` = `111 001 011`

PTE 1: `0xFF` at `0x0F`

PTE 1: PPN `111` (7) valid `1`

PTE 2: `0x0C` at `0x39`

PTE 2: PPN `000` (0) valid `0`

page fault!

2-level exercise (4)

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x08`; translate virtual address `0x1CB`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

0x1CB = 111 001 011
PTE 1: 0xFF at 0x0F
PTE 1: PPN 111 (7) valid 1
PTE 2: 0x0C at 0x39
PTE 2: PPN 000 (0) valid 0
page fault!

2-level exercise (5)

10-bit virtual addresses, 6-bit physical; 16 byte pages, 2 byte PTE

page tables 1 page; PTE: 2 bit PPN (MSB of first byte), 1 valid bit, rest unused

page table base register `0x10`; translate virtual address `0x376`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	AC BC DC EC

physical addresses	bytes
0x20–3	D0 E1 D2 D3
0x24–7	D4 E5 D6 E7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

2-level exercise (5)

10-bit virtual addresses, 6-bit physical; 16 byte pages, 2 byte PTE

page tables 1 page; PTE: 2 bit PPN (MSB of first byte), 1 valid bit, rest unused

page table base register `0x10`; translate virtual address `0x376`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	AC BC DC EC

physical addresses	bytes
0x20–3	D0 E1 D2 D3
0x24–7	D4 E5 D6 E7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x376` = `110 111 0110`

PTE 1: $0x10 + 6 \times 2 =$ `0x1C`: `AC BC`

PTE 1: PPN `10` valid `1`

PTE 2: $0x20 + 7 \times 2 =$ `0x2E`: `EF F0`

PTE 2: PPN `11` valid `1`

`11 0110` = `0x36` → DB

2-level exercise (5)

10-bit virtual addresses, 6-bit physical; 16 byte pages, 2 byte PTE

page tables 1 page; PTE: 2 bit PPN (MSB of first byte), 1 valid bit, rest unused

page table base register `0x10`; translate virtual address `0x376`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	AC BC DC EC

physical addresses	bytes
0x20–3	D0 E1 D2 D3
0x24–7	D4 E5 D6 E7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

0x376 = 110 111 0110

PTE 1: 0x10 + 6×2 = 0x1C: AC BC

PTE 1: PPN 10 valid 1

PTE 2: 0x20 + 7×2 = 0x2E: EF F0

PTE 2: PPN 11 valid 1

11 0110 = 0x36 → DB

2-level exercise (5)

10-bit virtual addresses, 6-bit physical; 16 byte pages, 2 byte PTE

page tables 1 page; PTE: 2 bit PPN (MSB of first byte), 1 valid bit, rest unused

page table base register `0x10`; translate virtual address `0x376`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	AC BC DC EC

physical addresses	bytes
0x20–3	D0 E1 D2 D3
0x24–7	D4 E5 D6 E7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x376` = `110 111 0110`

PTE 1: $0x10 + 6 \times 2 = 0x1C$: AC BC

PTE 1: PPN `10` valid `1`

PTE 2: $0x20 + 7 \times 2 = 0x2E$: EF F0

PTE 2: PPN `11` valid `1`

`11 0110` = `0x36` → DB

2-level exercise (5)

10-bit virtual addresses, 6-bit physical; 16 byte pages, 2 byte PTE

page tables 1 page; PTE: 2 bit PPN (MSB of first byte), 1 valid bit, rest unused

page table base register `0x10`; translate virtual address `0x376`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	AC BC DC EC

physical addresses	bytes
0x20–3	D0 E1 D2 D3
0x24–7	D4 E5 D6 E7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

0x376 = 110 111 0110

PTE 1: $0x10 + 6 \times 2 = 0x1C$: AC BC

PTE 1: PPN 10 valid 1

PTE 2: $0x20 + 7 \times 2 = 0x2E$: EF F0

PTE 2: PPN 11 valid 1

11 0110 = 0x36 → DB

2-level exercise (5)

10-bit virtual addresses, 6-bit physical; 16 byte pages, 2 byte PTE

page tables 1 page; PTE: 2 bit PPN (MSB of first byte), 1 valid bit, rest unused

page table base register `0x10`; translate virtual address `0x376`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	AC BC DC EC

physical addresses	bytes
0x20–3	D0 E1 D2 D3
0x24–7	D4 E5 D6 E7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

0x376 = 110 111 0110

PTE 1: $0x10 + 6 \times 2 = 0x1C$: AC BC

PTE 1: PPN 10 valid 1

PTE 2: $0x20 + 7 \times 2 = 0x2E$: EF F0

PTE 2: PPN 11 valid 1

11 0110 = 0x36 → DB

2-level exercise (5)

10-bit virtual addresses, 6-bit physical; 16 byte pages, 2 byte PTE

page tables 1 page; PTE: 2 bit PPN (MSB of first byte), 1 valid bit, rest unused

page table base register `0x10`; translate virtual address `0x376`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	AC BC DC EC

physical addresses	bytes
0x20–3	D0 E1 D2 D3
0x24–7	D4 E5 D6 E7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

0x376 = 110 111 0110

PTE 1: $0x10 + 6 \times 2 = 0x1C$: AC BC

PTE 1: PPN 10 valid 1

PTE 2: $0x20 + 7 \times 2 = 0x2E$: EF F0

PTE 2: PPN 11 valid 1

11 0110 = 0x36 → DB

2-level exercise (5)

10-bit virtual addresses, 6-bit physical; 16 byte pages, 2 byte PTE

page tables 1 page; PTE: 2 bit PPN (MSB of first byte), 1 valid bit, rest unused

page table base register `0x10`; translate virtual address `0x376`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	AC BC DC EC

physical addresses	bytes
0x20–3	D0 E1 D2 D3
0x24–7	D4 E5 D6 E7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

0x376 = 110 111 0110

PTE 1: $0x10 + 6 \times 2 = 0x1C$: AC BC

PTE 1: PPN 10 valid 1

PTE 2: $0x20 + 7 \times 2 = 0x2E$: EF F0

PTE 2: PPN 11 valid 1

11 0110 = 0x36 → DB

page table space exercise (1)

4-level page table

512 PTEs of 8 bytes each for each page table

suppose a process has exactly one page allocated

how much space for page tables?

page table space exercise (1)

4-level page table

512 PTEs of 8 bytes each for each page table

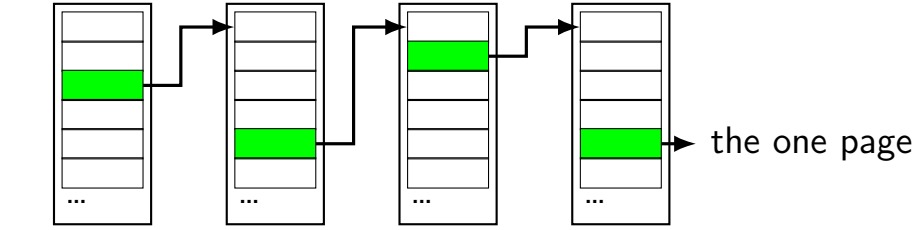
suppose a process has exactly one page allocated

how much space for page tables?

1 page at each level (4KB each)

exactly one valid entry in each of them

page table space exercise (1)



4 page tables at 1 page/page table
plus 1 page of data
5 pages total

page table space exercise (2)

4-level page table

512 PTEs of 8 bytes each for each page table

suppose a process has exactly two pages allocated:

one at address 0x0, one at address `0x20000000000`

how much space for page tables?

page table space exercise (2)

4-level page table

512 PTEs of 8 bytes each for each page table

suppose a process has exactly two pages allocated:
one at address 0x0, one at address `0x20000000000`

how much space for page tables?

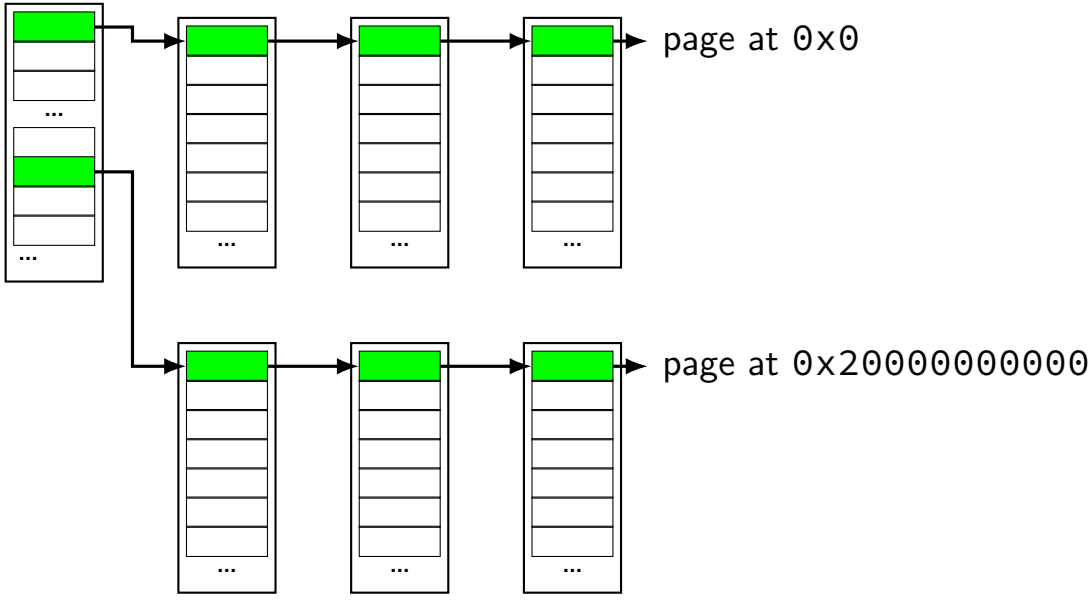
1 shared first-level PT, with two valid entries

two second-level PTs, each with one valid entry

two third-level PTs, each with one valid entry

two fourth-level PTs, each with one valid entry

page table space exercise (2)



page table space exercise (3)

4-level page table; each PT: 512 PTEs of 8 bytes

suppose a process has 100 pages of stack, 100 pages of code+constants (contiguous)

- stack and code+constants far apart

how much space for page tables?

page table space exercise (3)

4-level page table; each PT: 512 PTEs of 8 bytes

suppose a process has 100 pages of stack, 100 pages of code+constants (contiguous)

stack and code+constants far apart

how much space for page tables? — *minimum:*

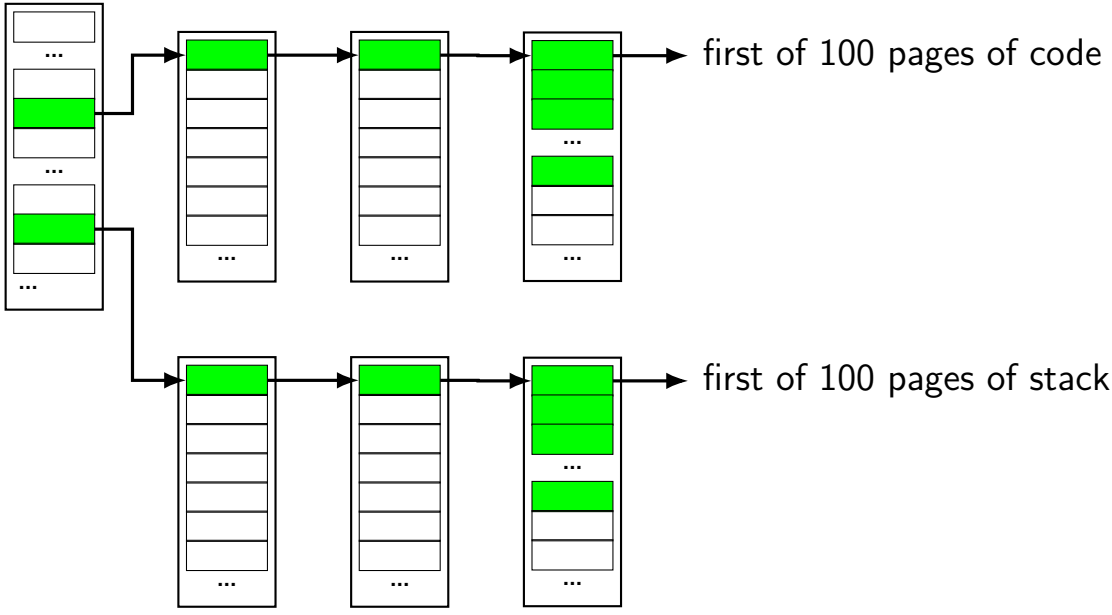
1 shared first-level PT, with two valid entries

two second-level PT, each with one valid entry

two third-level PT, each with one valid entry

two fourth-level PT, each with 100 valid entries

page table space exercise (3)



page table space exercise (3)

4-level page table; each PT: 512 PTEs of 8 bytes

suppose a process has 100 pages of stack, 100 pages of code+constants (contiguous)

how much space for page tables?

page table space exercise (3)

4-level page table; each PT: 512 PTEs of 8 bytes

suppose a process has 100 pages of stack, 100 pages of code+constants (contiguous)

how much space for page tables? — *maximum:*

1 shared first-level PT, with four valid entries

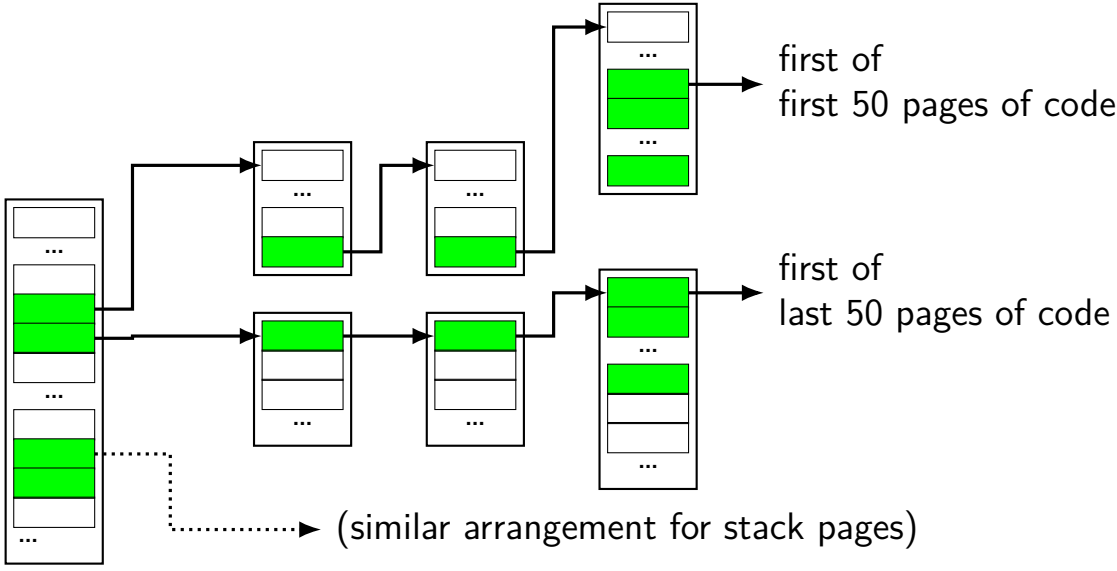
four second-level PT, each with one valid entry

two for stack, two for code+constants

four third-level PT, each with one valid entry

four fourth-level PT, each with 50 valid entries

page table space exercise (3)



page table space exercise (4)

4-level page table; each PT: 512 PTEs of 8 bytes

suppose a process has 200 pages, randomly distributed in PT

about how much space for page tables?

page table space exercise (4)

4-level page table; each PT: 512 PTEs of 8 bytes

suppose a process has 200 pages, randomly distributed in PT

about how much space for page tables?

about 165 ($\pm \sim 8$) entries in first-level PT

(some pages randomly share first-level PT entries)

about 165 second-level PTs, 200 third-level, 200 fourth-level

a bit less than 600 page tables — almost 2400 KB

cache accesses and multi-level PTs

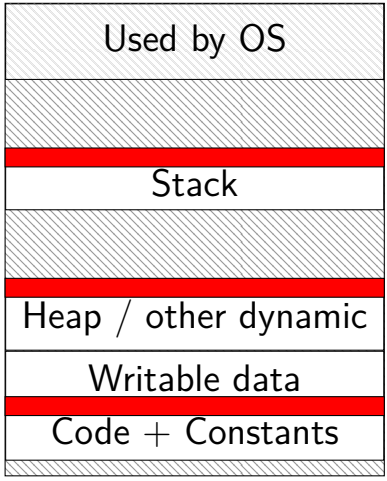
four-level page tables — four cache accesses per memory access

L1 cache hits — typically a couple cycles each?

so add 8 cycles to each memory access?

not acceptable

program memory active sets



0xFFFF FFFF FFFF FFFF

0xFFFF 8000 0000 0000

0x7F...

small areas of memory active at a time
one or two pages in each area?

0x0000 0000 0040 0000

page table entries and locality

page table entries have excellent temporal locality

typically one or two pages of the stack active

typically one or two pages of code active

typically one or two pages of heap/globals active

each page contains whole functions, arrays, stack frames, etc.

page table entries and locality

page table entries have excellent temporal locality

typically one or two pages of the stack active

typically one or two pages of code active

typically one or two pages of heap/globals active

each page contains whole functions, arrays, stack frames, etc.

needed page table entries are very small

page table entry cache

caled a **TLB** (translation lookaside buffer)

very small cache of page table entries

L1 cache	TLB
physical addresses	virtual page numbers
bytes from memory	page table entries
tens of bytes per block	one page table entry per block
usually thousands of blocks	usually tens of entries

page table entry cache

caled a **TLB** (translation lookaside buffer)

very small cache of page table entries

L1 cache	TLB
physical addresses	virtual page numbers
bytes from memory	page table entries
tens of bytes per block	one page table entry per block
usually thousands of blocks	usually tens of entries

only caches the page table lookup itself
(generally) just entries from the last-level page table

page table entry cache

caled a **TLB** (translation lookaside buffer)

very small cache of page table entries

L1 cache	TLB
physical addresses	virtual page numbers
bytes from memory	page table entries
tens of bytes per block	one page table entry per block
usually thousands of blocks	usually tens of entries

not much spatial locality between page table entries
(they're used for kilobytes of data already)

page table entry cache

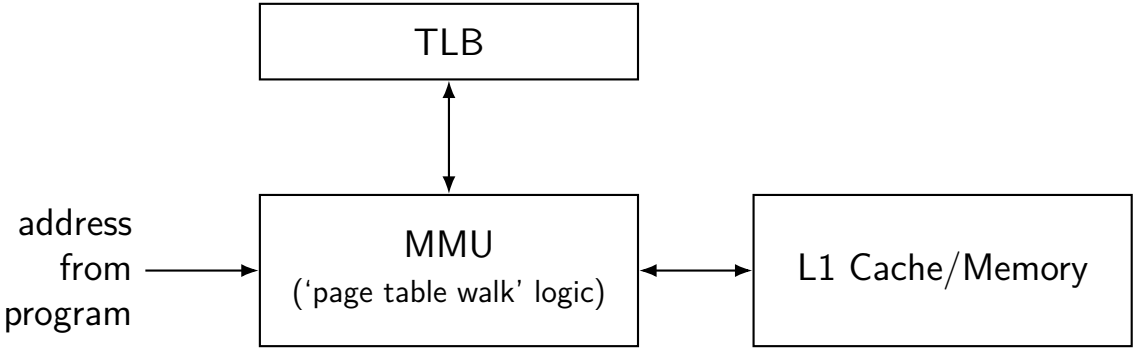
caled a **TLB** (translation lookaside buffer)

very small cache of page table entries

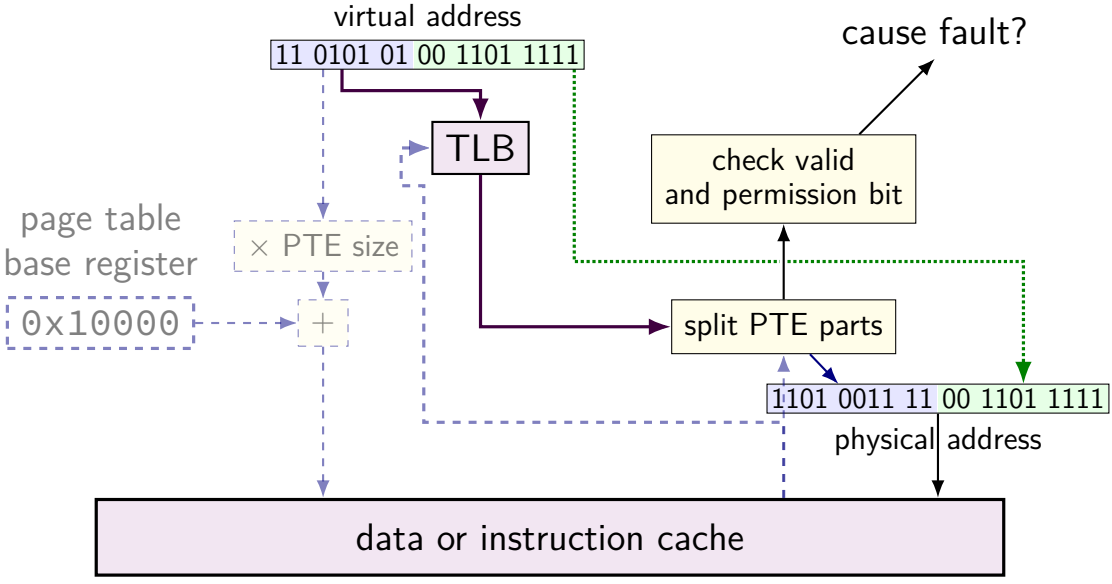
L1 cache	TLB
physical addresses	virtual page numbers
bytes from memory	page table entries
tens of bytes per block	one page table entry per block
usually thousands of blocks	usually tens of entries

few active page table entries at a time
enables highly associative cache designs

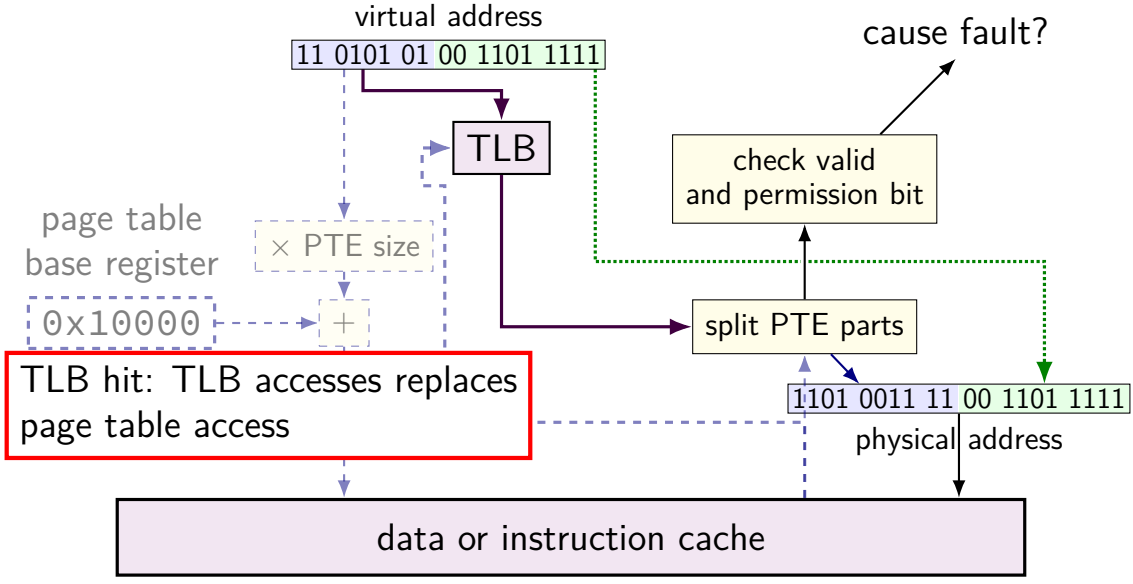
TLB and the MMU (1)



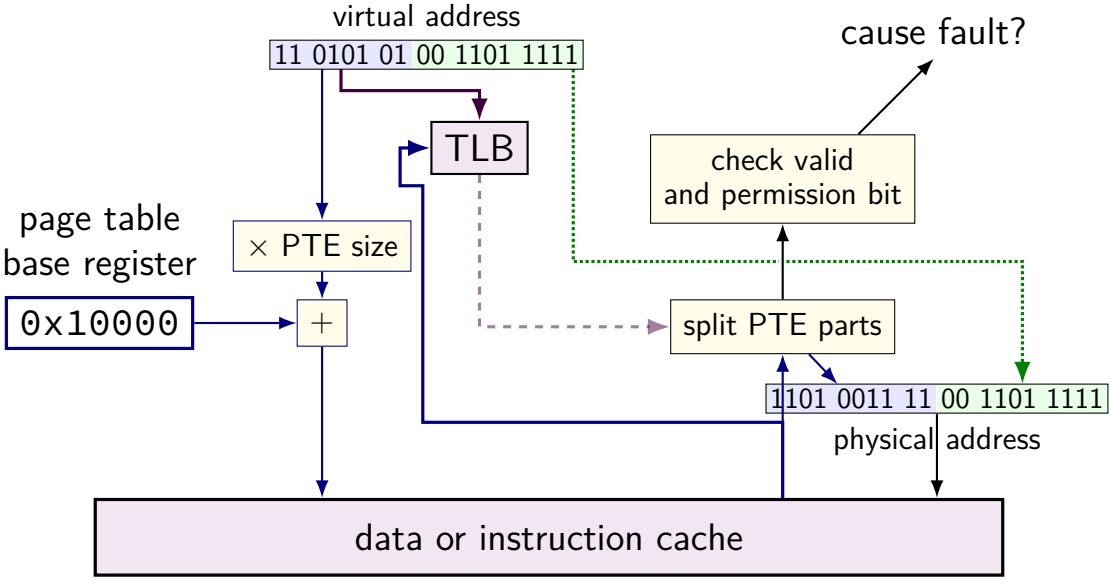
TLB and the MMU (2)



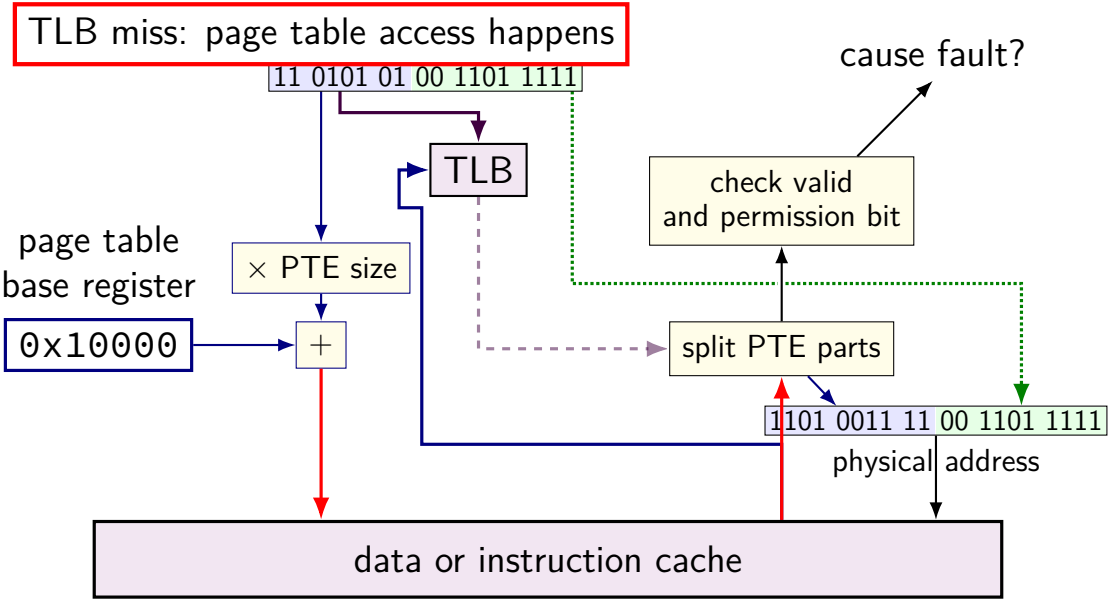
TLB and the MMU (2)



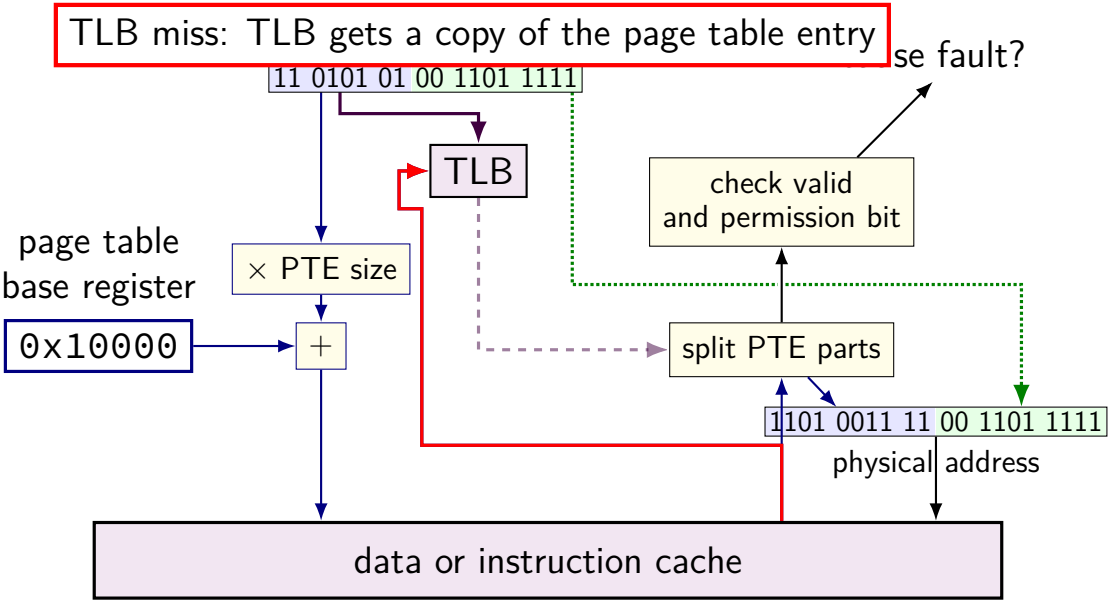
TLB and the MMU (2)



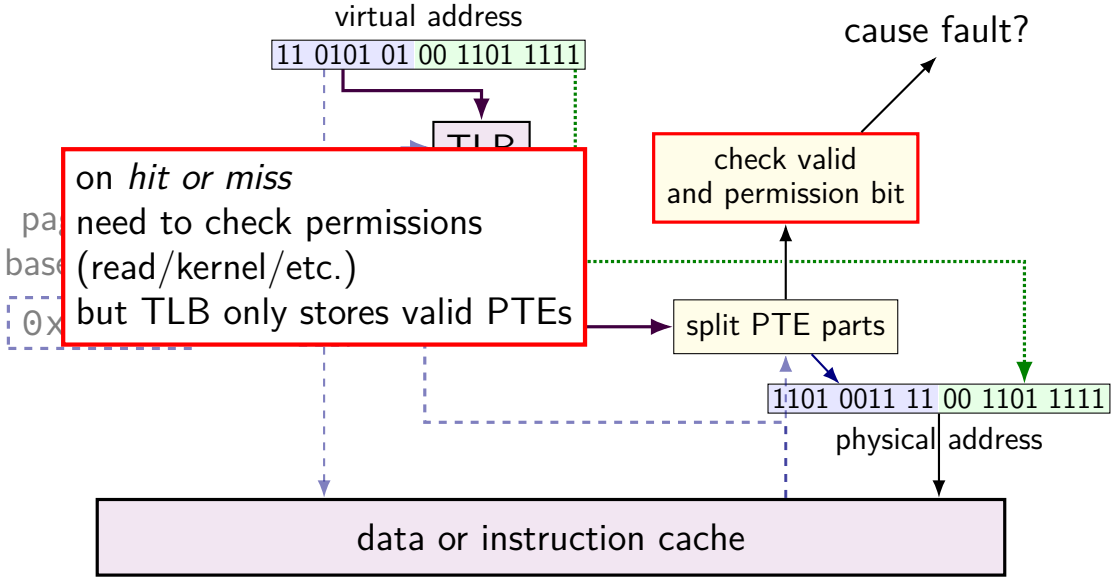
TLB and the MMU (2)



TLB and the MMU (2)



TLB and the MMU (2)



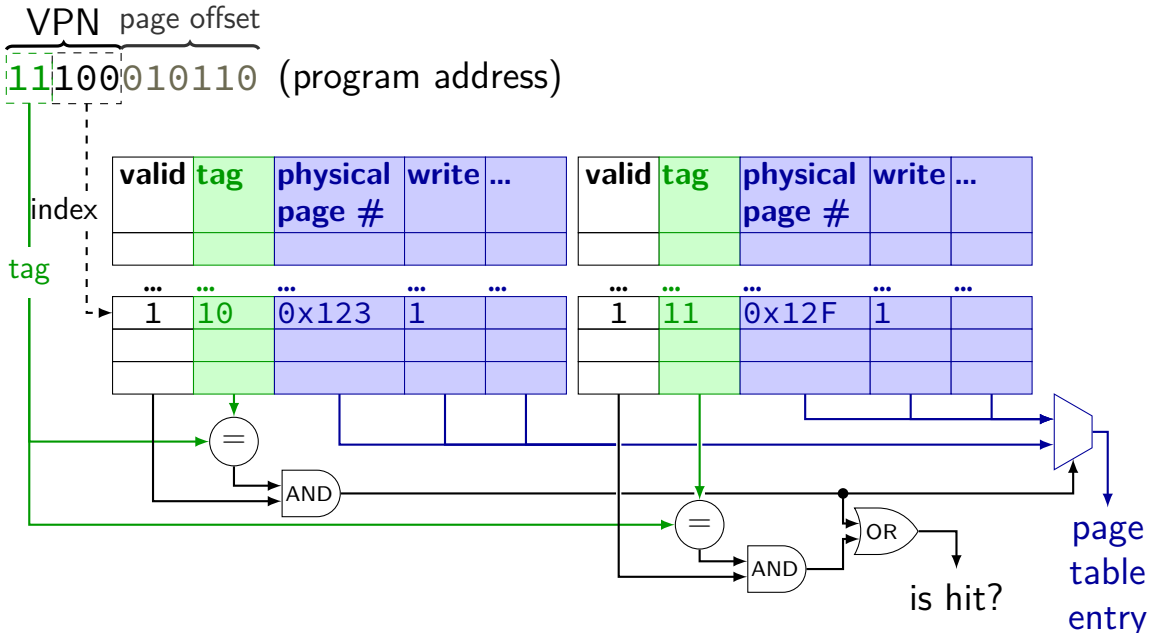
TLB and multi-level page tables

TLB caches valid last-level page table entries

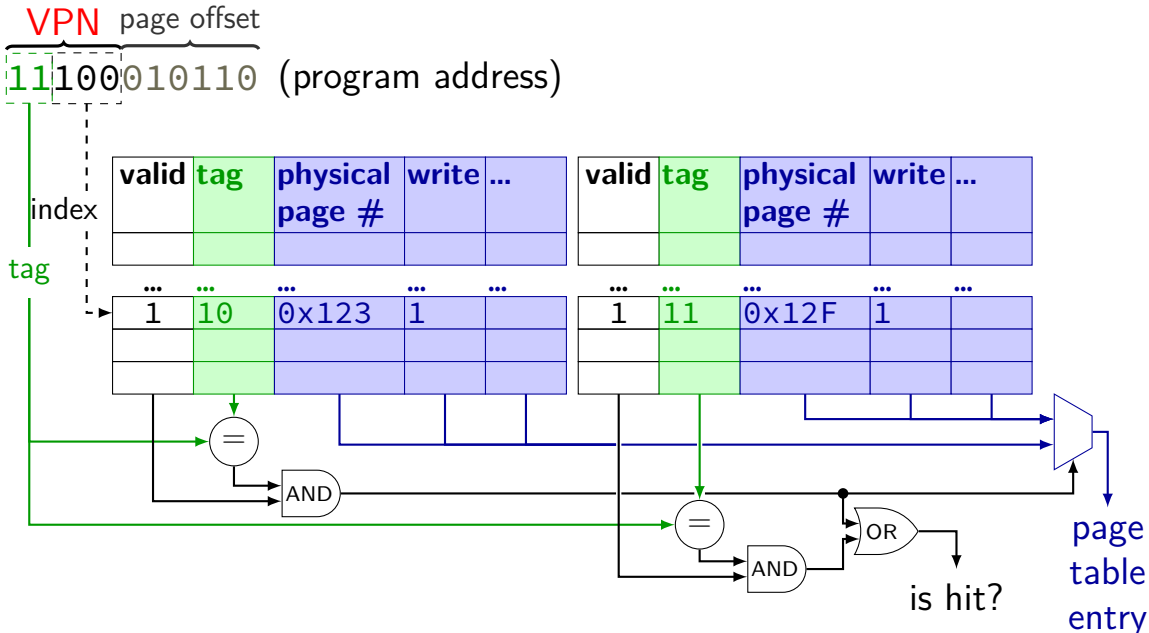
doesn't matter which last-level page table

means TLB output can be used directly to form address

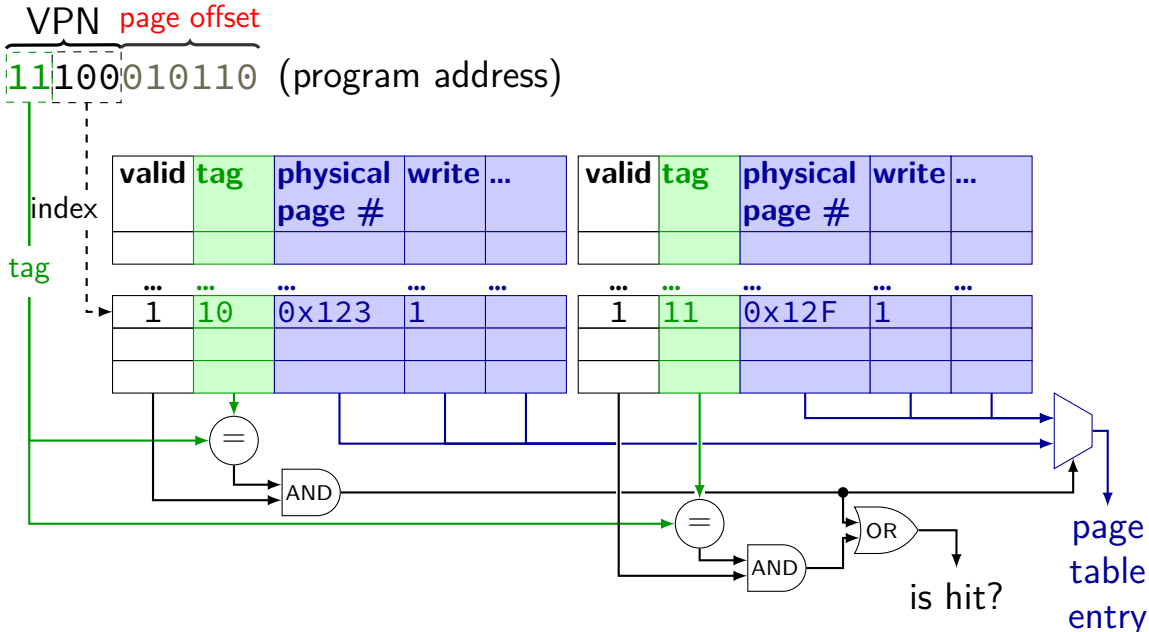
TLB organization (2-way set associative)



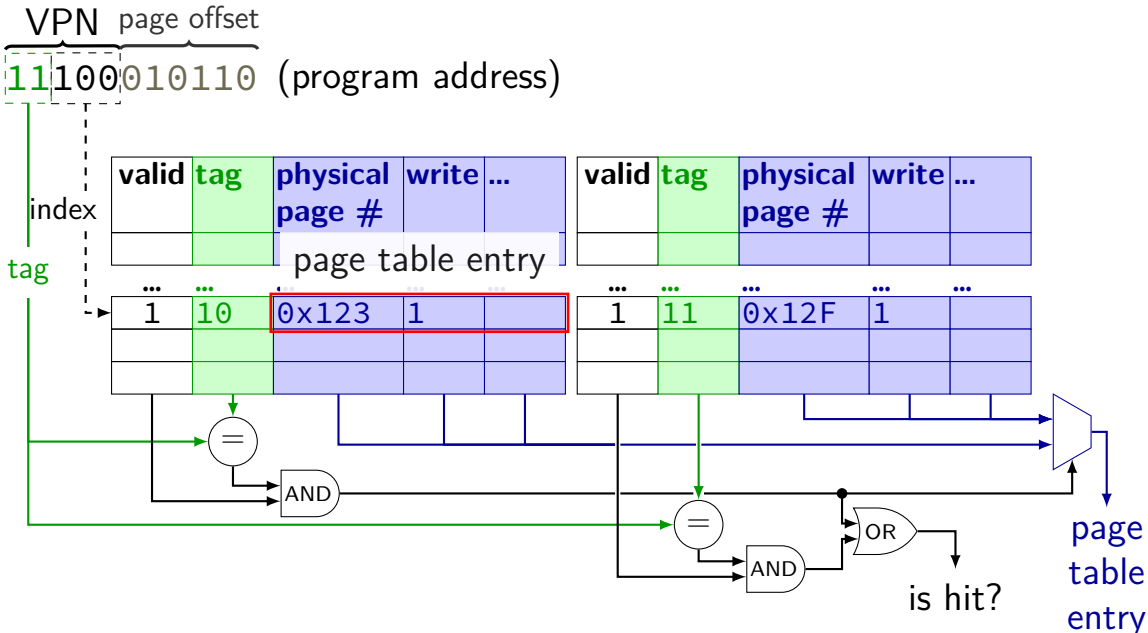
TLB organization (2-way set associative)



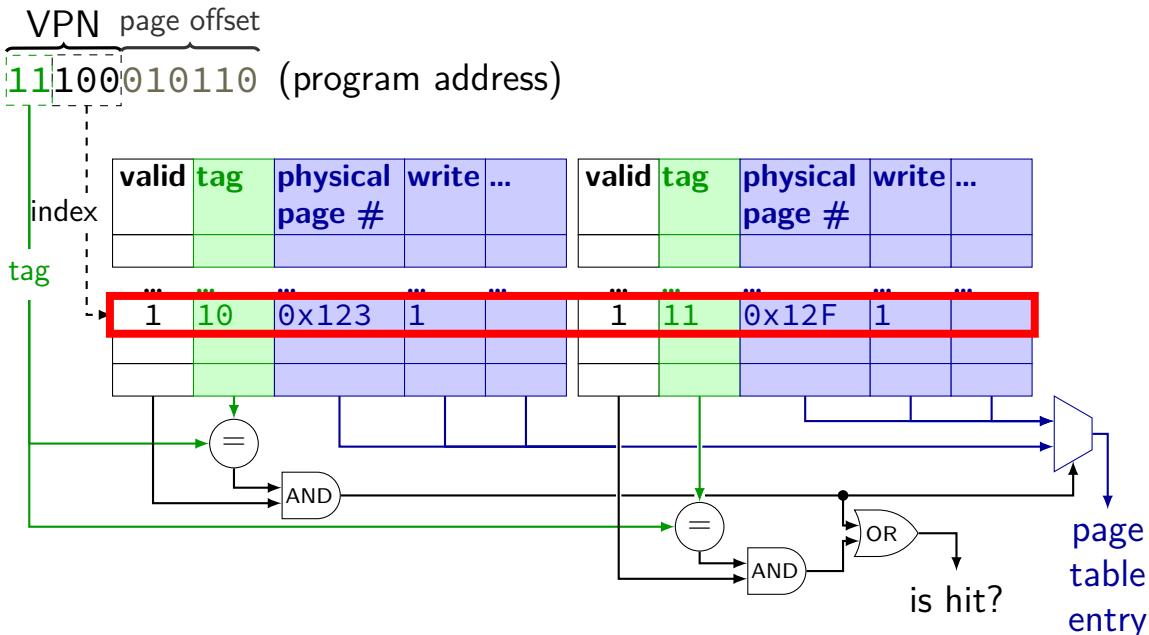
TLB organization (2-way set associative)



TLB organization (2-way set associative)



TLB organization (2-way set associative)



address splitting for TLBs (1)

my desktop:

4KB (2^{12} byte) pages; 48-bit virtual address

64-entry, 4-way L1 data TLB

TLB index bits?

TLB tag bits?

address splitting for TLBs (1)

my desktop:

4KB (2^{12} byte) pages; 48-bit virtual address

64-entry, 4-way L1 data TLB

TLB index bits?

$64/4 = 16$ sets — 4 bits

TLB tag bits?

$48 - 12 = 36$ bit virtual page number — $36 - 4 = 32$ bit TLB tag

address splitting for TLBs (2)

my desktop:

4KB (2^{12} byte) pages; 48-bit virtual address

1536-entry ($3 \cdot 2^9$), 12-way L2 TLB

TLB index bits?

TLB tag bits?

address splitting for TLBs (2)

my desktop:

4KB (2^{12} byte) pages; 48-bit virtual address

1536-entry ($3 \cdot 2^9$), 12-way L2 TLB

TLB index bits?

$1536/12 = 128$ sets — 7 bits

TLB tag bits?

$48 - 12 = 36$ bit virtual page number — $36 - 7 = 29$ bit TLB tag

address splitting exercise (3)

384-entry, 3-way set-associative TLB

32-bit virtual address; 8KB pages

2-level page table; 4 byte PTEs

256 entries in first level; 2048 in second

split the address `0x12345678`

address splitting exercise (3)

384-entry, 3-way set-associative TLB

32-bit virtual address; 8KB pages

2-level page table; 4 byte PTEs

- 256 entries in first level; 2048 in second

split the address `0x12345678`

```
0001 0010 0011 0100 0101 0110 0111 1000
```

address splitting exercise (3)

384-entry, 3-way set-associative TLB

32-bit virtual address; 8KB pages

2-level page table; 4 byte PTEs

- 256 entries in first level; 2048 in second

split the address `0x12345678`

- `0001 0010 0011 0100 0101 0110 0111 1000`
- 13-bit page offset `1 0110 0111 1000`

address splitting exercise (3)

384-entry, 3-way set-associative TLB

32-bit virtual address; 8KB pages

2-level page table; 4 byte PTEs

- 256 entries in first level; 2048 in second

split the address `0x12345678`

- `0001 0010 0011 0100 0101 0110 0111 1000`
- 13-bit page offset `1 0110 0111 1000`
- $32 - 13 = 19$-bit VPN `0001 0010 0011 0100 010`

address splitting exercise (3)

384-entry, 3-way set-associative TLB

32-bit virtual address; 8KB pages

2-level page table; 4 byte PTEs

- 256 entries in first level; 2048 in second

split the address `0x12345678`

- `0001 0010 0011 0100 0101 0110 0111 1000`
- 13-bit page offset `1 0110 0111 1000`
- $32 - 13 = 19$-bit VPN `0001 0010 0011 0100 010`
- 8-bit first part of VPN `0001 0010`

address splitting exercise (3)

384-entry, 3-way set-associative TLB

32-bit virtual address; 8KB pages

2-level page table; 4 byte PTEs

- 256 entries in first level; 2048 in second

split the address `0x12345678`

- `0001 0010 0011 0100 0101 0110 0111 1000`
- 13-bit page offset `1 0110 0111 1000`
- $32 - 13 = 19$-bit VPN `0001 0010 0011 0100 010`
- 8-bit first part of VPN `0001 0010`
- 11-bit second part of VPN `0011 0100 010`

address splitting exercise (3)

384-entry, 3-way set-associative TLB

32-bit virtual address; 8KB pages

2-level page table; 4 byte PTEs

- 256 entries in first level; 2048 in second

split the address `0x12345678`

- `0001 0010 0011 0100 0101 0110 0111 1000`
- 13-bit page offset `1 0110 0111 1000`
- $32 - 13 = 19$-bit VPN `0001 0010 0011 0100 010`
- 8-bit first part of VPN `0001 0010`
- 11-bit second part of VPN `0011 0100 010`
- 7-bit TLB index 0100 010

address splitting exercise (3)

384-entry, 3-way set-associative TLB

32-bit virtual address; 8KB pages

2-level page table; 4 byte PTEs

- 256 entries in first level; 2048 in second

split the address `0x12345678`

- `0001 0010 0011 0100 0101 0110 0111 1000`
- 13-bit page offset `1 0110 0111 1000`
- $32 - 13 = 19$-bit VPN `0001 0010 0011 0100 010`
- 8-bit first part of VPN `0001 0010`
- 11-bit second part of VPN `0011 0100 010`
- 7-bit TLB index 0100 010
- $19 - 7 = 12$-bit TLB tag 0001 0010 0011

exercise: TLB access pattern (setup)

4-entry, 2-way TLB, LRU replacement policy, initially empty

4096 byte pages

how many index bits?

TLB index of virtual address 0x12345?

backup slides

splitting addresses for levels

x86-32

32-bit physical address; 32-bit virtual address

2^{12} byte page size

2-levels of page tables; each page table is one page

4 byte page table entries

how is address `0x12345678` split up?

splitting addresses for levels

x86-32

32-bit physical address; 32-bit virtual address

2^{12} byte page size

12-bit page offset

2-levels of page tables; each page table is one page

4 byte page table entries

how is address `0x12345678` split up?

splitting addresses for levels

x86-32

32-bit physical address; 32-bit virtual address

2^{12} byte page size

12-bit page offset

2-levels of page tables; each page table is one page

4 byte page table entries

$2^{12}/4 = 2^{10}$ *PTEs/page table; 10-bit VPN parts*

how is address `0x12345678` split up?

splitting addresses for levels

x86-32

32-bit physical address; 32-bit virtual address

2^{12} byte page size

12-bit page offset

2-levels of page tables; each page table is one page

4 byte page table entries

$2^{12}/4 = 2^{10}$ *PTEs/page table; 10-bit VPN parts*

how is address `0x12345678` split up?

10-bit VPN part 1: `0001 0010 00 (0x48);`

10-bit VPN part 2: `11 0100 0101 (0x345);`

12-bit page offset: `0x678`

1-level example

6-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 other bits;

page table base register `0x20`; translate virtual address `0x30`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	CB 0B CB 0B
0x38–B	DC 0C DC 0C
0x3C–F	EC 0C EC 0C

1-level example

6-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 other bits;

page table base register `0x20`; translate virtual address `0x30`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	CB 0B CB 0B
0x38–B	DC 0C DC 0C
0x3C–F	EC 0C EC 0C

0x30 = 11 0000

PTE addr:

0x20 + 6 ×1 = 0x26

PTE value:

0xD6 = 1101 0110

PPN 110, valid 1

M[110 000] = **M[**0x30]

1-level example

6-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 other bits;

page table base register `0x20`; translate virtual address `0x30`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	CB 0B CB 0B
0x38–B	DC 0C DC 0C
0x3C–F	EC 0C EC 0C

$0x30 = 11\ 0000$

PTE addr:

$0x20 + 6 \times 1 = 0x26$

PTE value:

$0xD6 = 1101\ 0110$

PPN 110, valid 1

$M[110\ 000] = \mathbf{M[}0x30]$

1-level example

6-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 other bits;

page table base register `0x20`; translate virtual address `0x30`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	CB 0B CB 0B
0x38–B	DC 0C DC 0C
0x3C–F	EC 0C EC 0C

0x30 = 11 0000

PTE addr:

0x20 + 6 ×1 = 0x26

PTE value:

0xD6 = 1101 0110

PPN 110, valid 1

M[110 000] = **M[**0x30]

2-level example

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x20`; translate virtual address `0x131`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

2-level example

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x20`; translate virtual address `0x131`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

0x131 = 1 0011 0001

0x20 + 4 ×1 = 0x24

PTE 1 value:

0xD4 = 1101 0100

PPN 110, valid 1

2-level example

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x20`; translate virtual address `0x131`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x131` = `1 00 11 00 01`

`0x20` + 4 × 1 = `0x24`

PTE 1 value:

`0xD4` = `1101 0100`

PPN `110`, valid `1`

PTE 2 addr:

`110 000` + `110` = `0x36`

PTE 2 value: `0xDB`

2-level example

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x20`; translate virtual address `0x131`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x131` = `1 0011 0001`

`0x20` + 4 × 1 = `0x24`

PTE 1 value:

`0xD4` = `1101 0100`

PPN `110`, valid `1`

PTE 2 addr:

`110 000` + `110` = `0x36`

PTE 2 value: `0xDB`

PPN `110`; valid `1`

M[`110 001` (`0x31`)] = `0x0A`

2-level example

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x20`; translate virtual address `0x131`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x131` = `1 0011 0001`

`0x20` + 4 × 1 = `0x24`

PTE 1 value:

`0xD4` = `1101 0100`

PPN `110`, valid `1`

PTE 2 addr:

`110 000` + `110` = `0x36`

PTE 2 value: `0xDB`

PPN `110`; valid `1`

M[`110 001` (`0x31`)] = `0x0A`

2-level example

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register `0x20`; translate virtual address `0x131`

physical addresses	bytes
0x00–3	00 11 22 33
0x04–7	44 55 66 77
0x08–B	88 99 AA BB
0x0C–F	CC DD EE FF
0x10–3	1A 2A 3A 4A
0x14–7	1B 2B 3B 4B
0x18–B	1C 2C 3C 4C
0x1C–F	1C 2C 3C 4C

physical addresses	bytes
0x20–3	D0 D1 D2 D3
0x24–7	D4 D5 D6 D7
0x28–B	89 9A AB BC
0x2C–F	CD DE EF F0
0x30–3	BA 0A BA 0A
0x34–7	DB 0B DB 0B
0x38–B	EC 0C EC 0C
0x3C–F	FC 0C FC 0C

`0x131` = `1 0011 0001`

`0x20` + 4 × `1` = `0x24`

PTE 1 value:

`0xD4` = `1101 0100`

PPN `110`, valid `1`

PTE 2 addr:

`110 000` + `110` = `0x36`

PTE 2 value: `0xDB`

PPN `110`; valid `1`

M[`110 001` (`0x31`)] = `0x0A`

2-level splitting

9-bit virtual address

6-bit physical address

8-byte pages $\rightarrow$ 3-bit page offset (bottom bits)

9-bit VA: 6 bit VPN + 3 bit PO

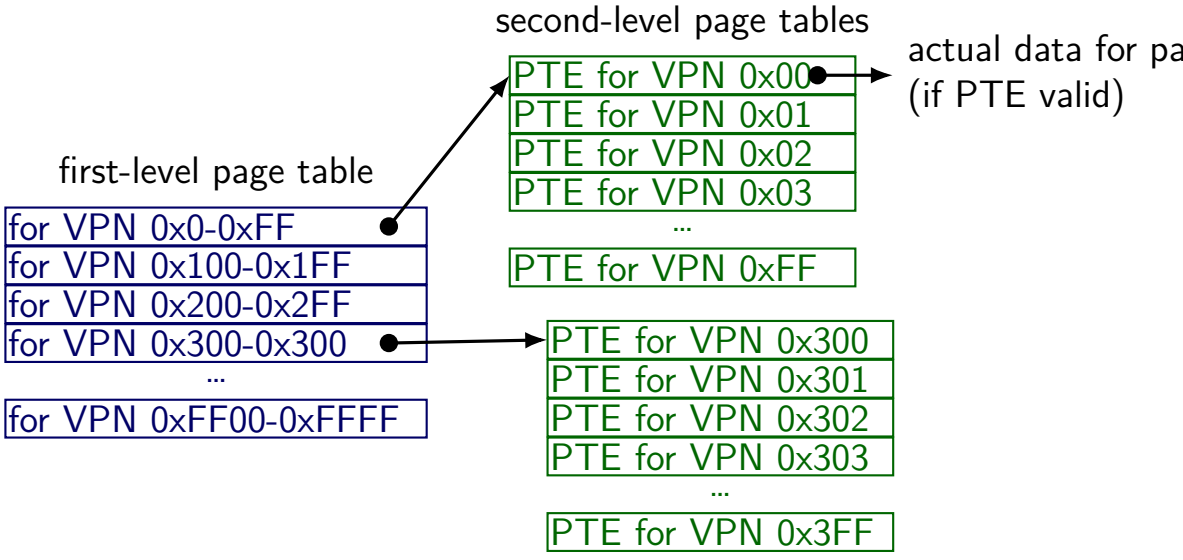
6-bit PA: 3 bit PPN + 3 bit PO

8 entry page tables $\rightarrow$ 3-bit VPN parts

9-bit VA: 3 bit VPN part 1; 3 bit VPN part 2

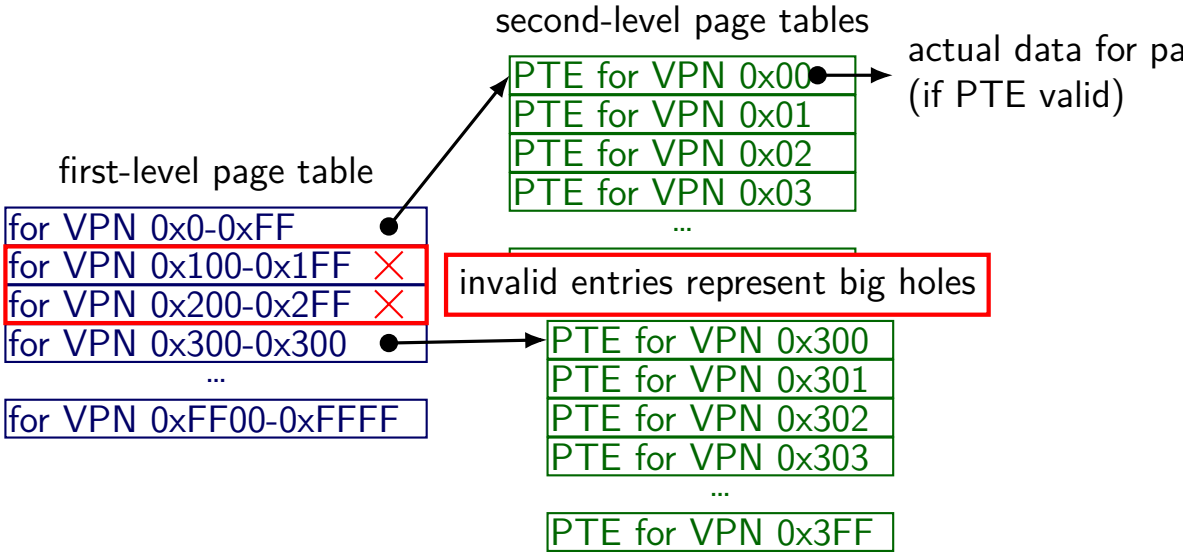
two-level page tables

two-level page table for 65536 pages (16-bit VPN)



two-level page tables

two-level page table for 65536 pages (16-bit VPN)



two-level page tables

two-level page table for 65536 pages (16-bit VPN)

first-level pag

for VPN 0x0-0xF

for VPN 0x100-0

for VPN 0x200-0

for VPN 0x300-0

...

for VPN 0xFF00

first-level page table

VPN range	valid	kernel	write	physical page # (of next page table)
0x0000–0x00FF	1	0	1	0x22343
0x0100–0x01FF	0	0	1	0x00000
0x0200–0x02FF	0	0	0	0x00000
0x0300–0x03FF	1	1	0	0x33454
0x0400–0x04FF	1	1	0	0xFF043
...	...	...	...	...
0xFF00–0xFFFF	1	1	0	0xFF045

PTE for VPN 0x303

...

PTE for VPN 0x3FF

two-level page tables

two-level page table for 65536 pages (16-bit VPN)

first-level pag

for VPN 0x0-0xF

for VPN 0x100-0

for VPN 0x200-0

for VPN 0x300-0

...

for VPN 0xFF00

first-level page table

VPN range	valid	kernel	write	physical page # (of next page table)
0x0000–0x00FF	1	0	1	0x22343
0x0100–0x01FF	0	0	1	0x00000
0x0200–0x02FF	0	0	0	0x00000
0x0300–0x03FF	1	1	0	0x33454
0x0400–0x04FF	1	1	0	0xFF043
...	...	...	...	...
0xFF00–0xFFFF	1	1	0	0xFF045

PTE for VPN 0x303

...

PTE for VPN 0x3FF

two-level page tables

two-level page table for 65536 pages (16-bit VPN)

first-level pag

for VPN 0x0-0xF

for VPN 0x100-0

for VPN 0x200-0

for VPN 0x300-0

...

for VPN 0xFF00

first-level page table

VPN range	valid	kernel	write	physical page # (of next page table)
0x0000-0x00FF	1	0	1	0x22343
0x0100-0x01FF	0	0	1	0x00000
0x0200-0x02FF	0	0	0	0x00000
0x0300-0x03FF	1	1	0	0x33454
0x0400-0x04FF	1	1	0	0xFF043
...	...	...	...	...
0xFF00-0xFFFF	1	1	0	0xFF045

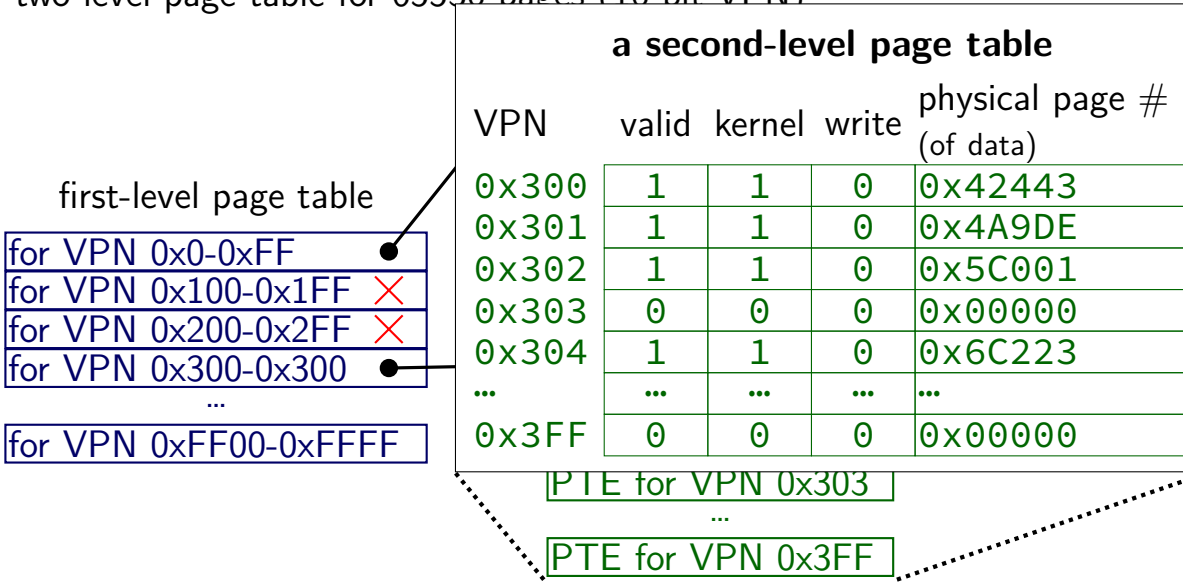
PTE for VPN 0x303

...

PTE for VPN 0x3FF

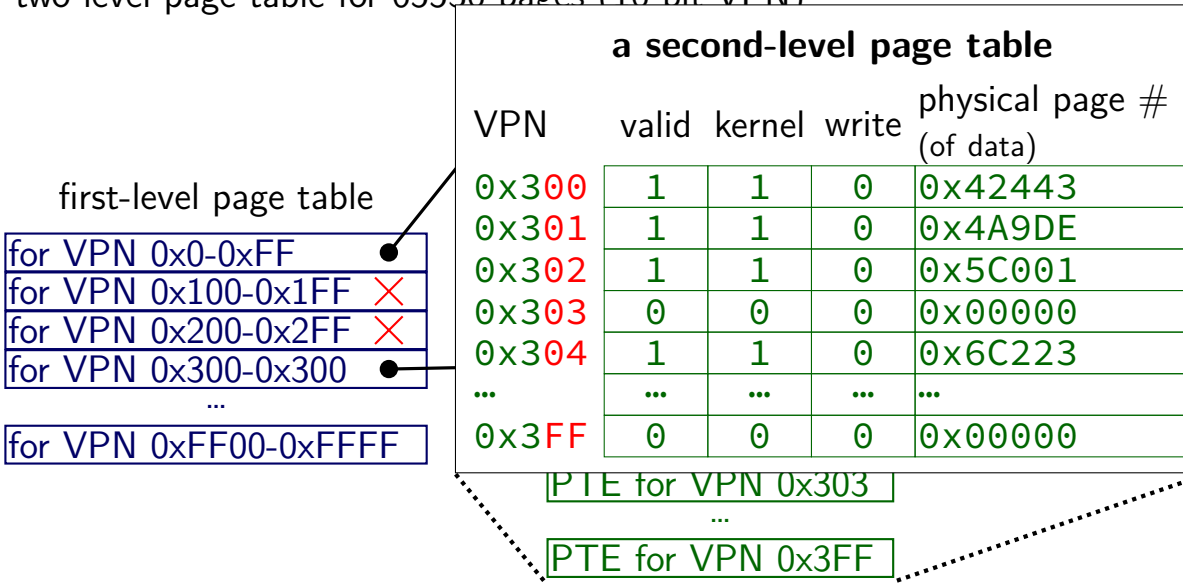
two-level page tables

two-level page table for 65536 pages (16-bit VPN)



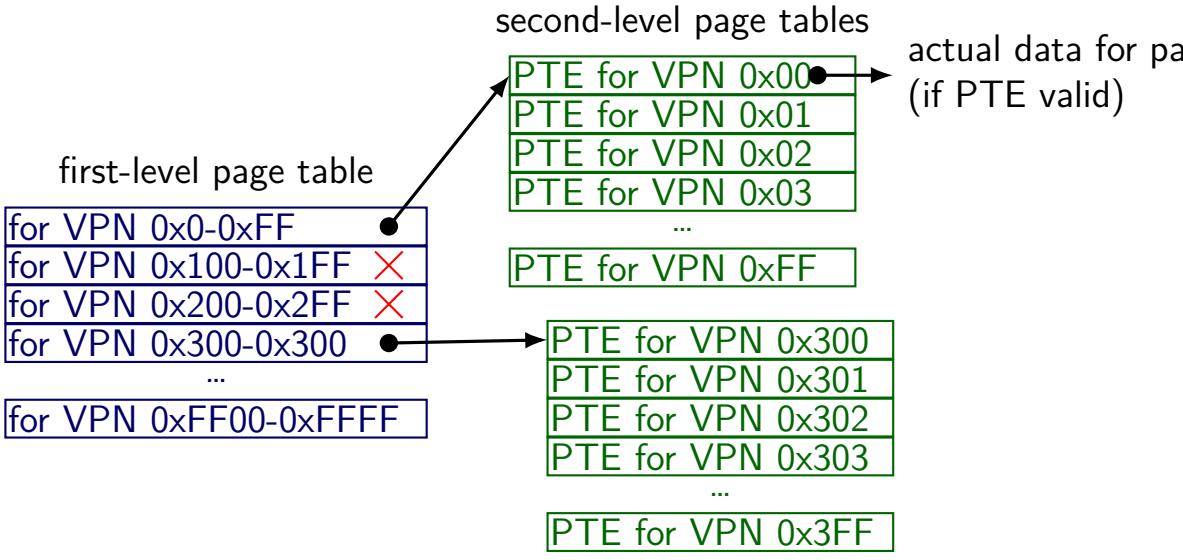
two-level page tables

two-level page table for 65536 pages (16-bit VPN)

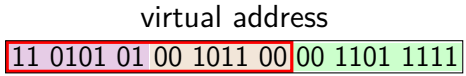


two-level page tables

two-level page table for 65536 pages (16-bit VPN)

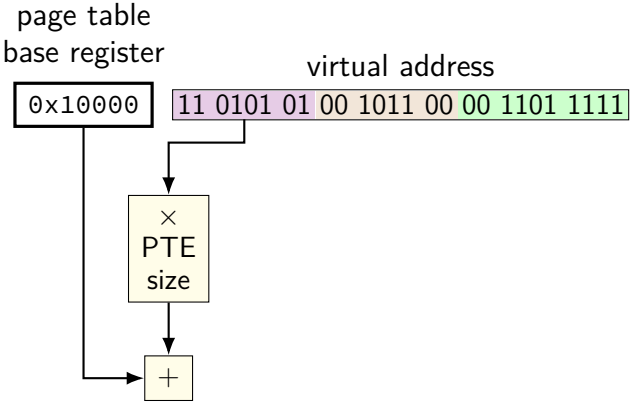


two-level page table lookup

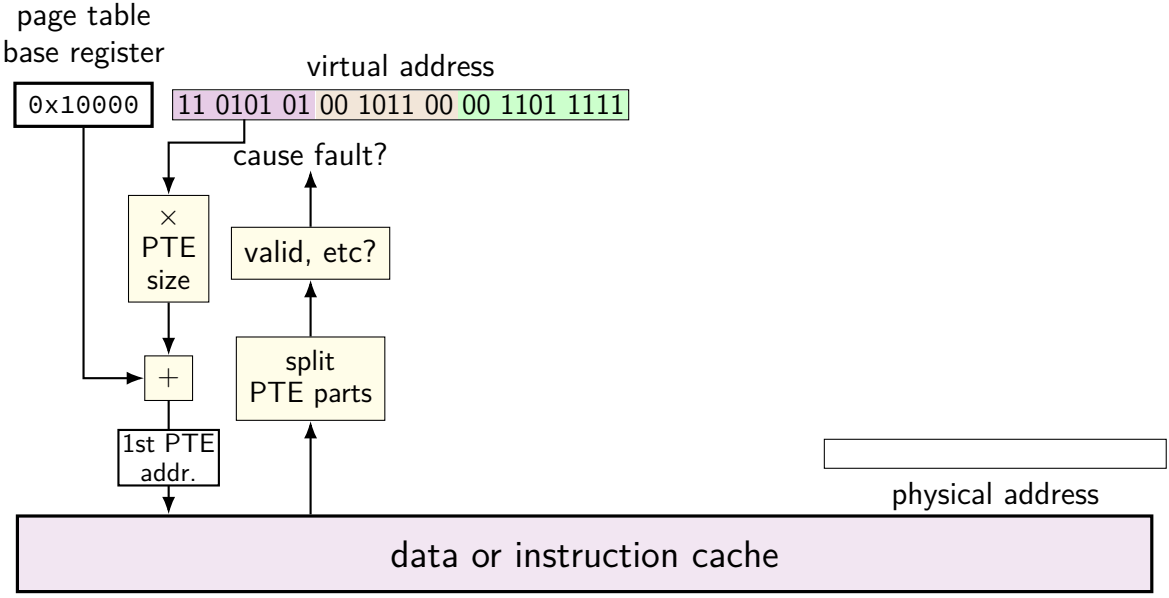


VPN — split into two parts (one per level)

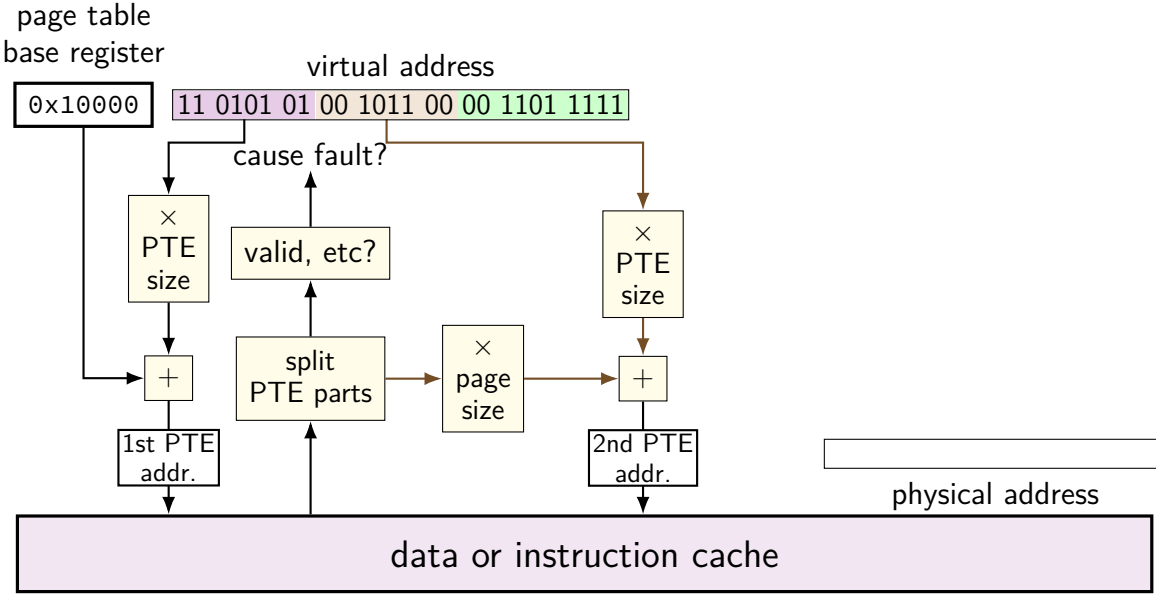
two-level page table lookup



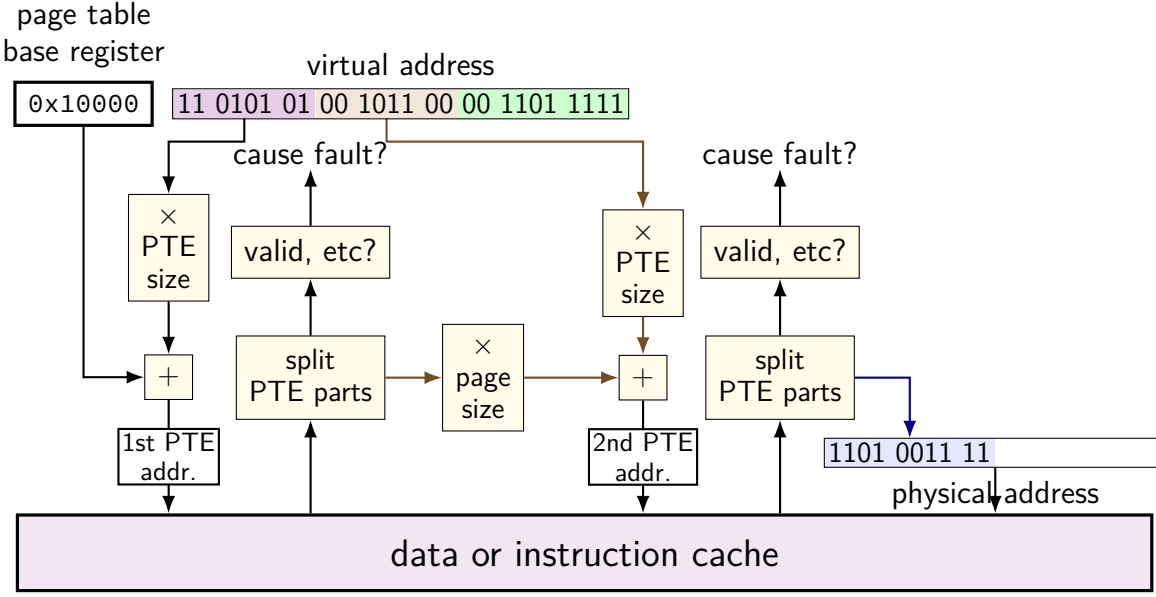
two-level page table lookup



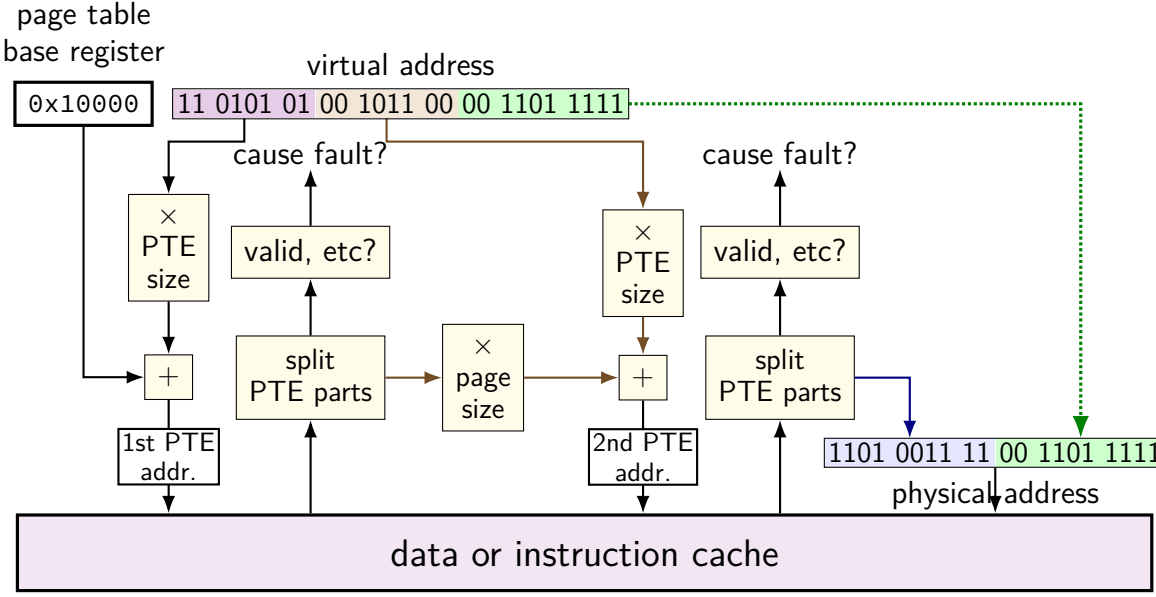
two-level page table lookup



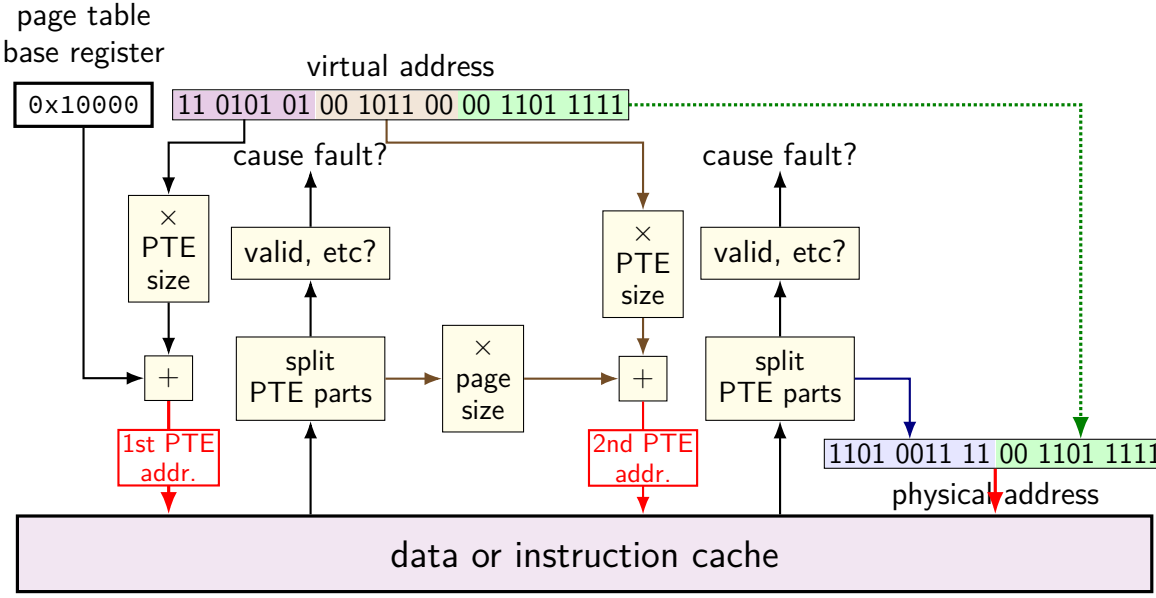
two-level page table lookup



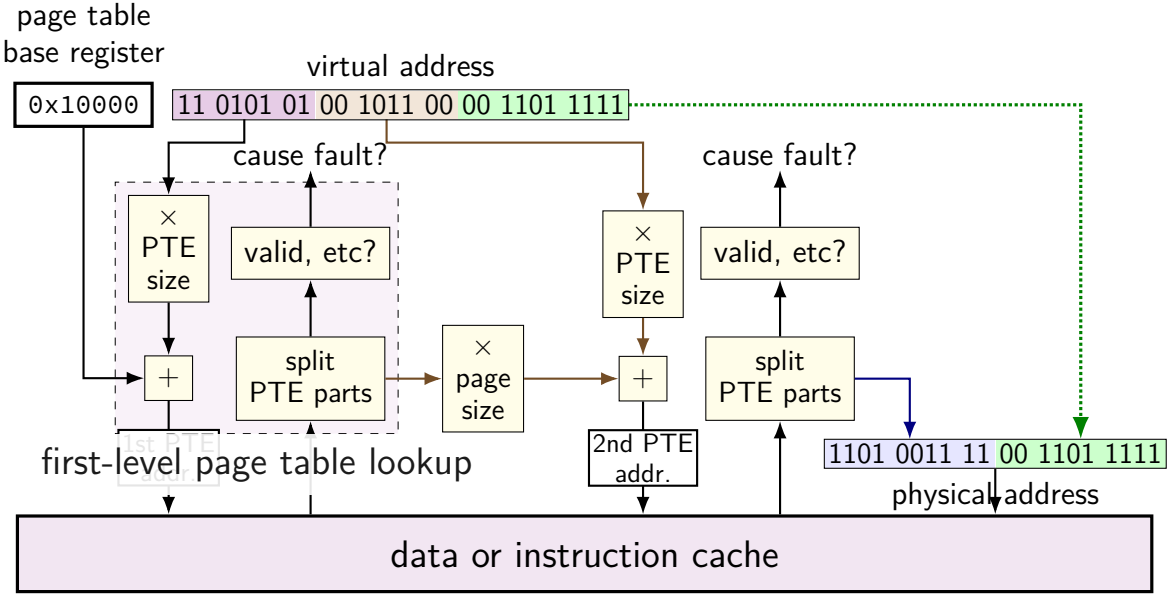
two-level page table lookup



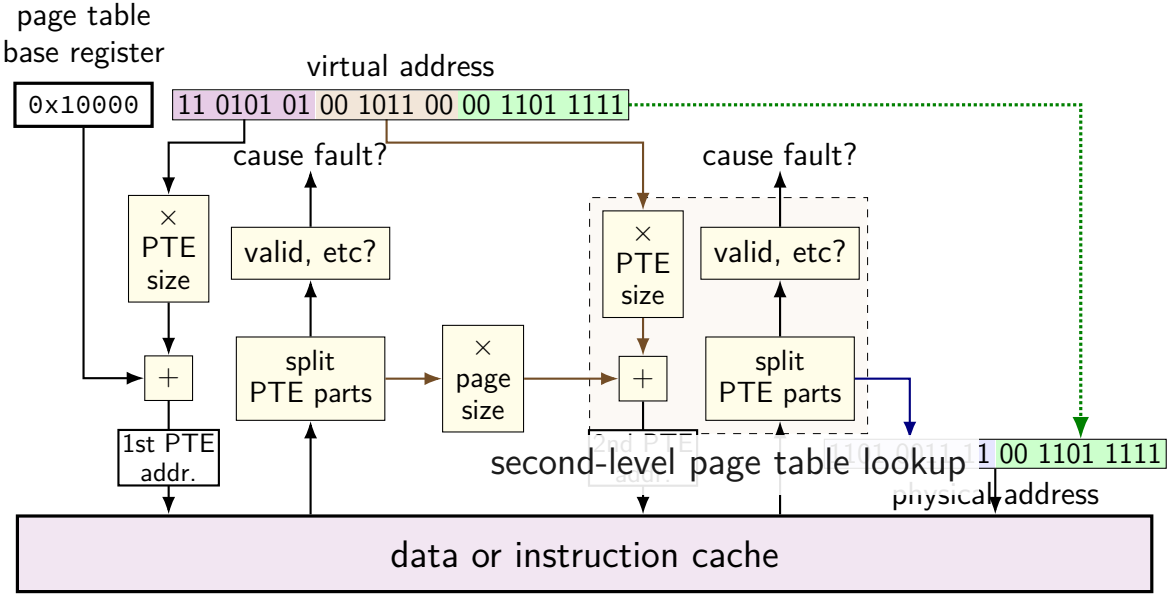
two-level page table lookup



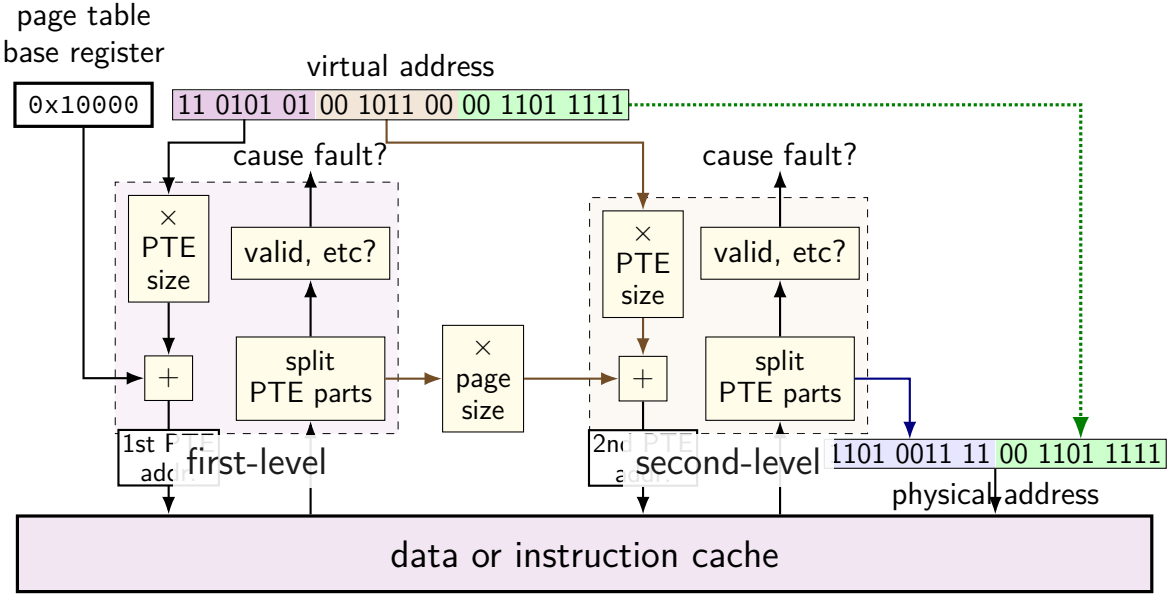
two-level page table lookup



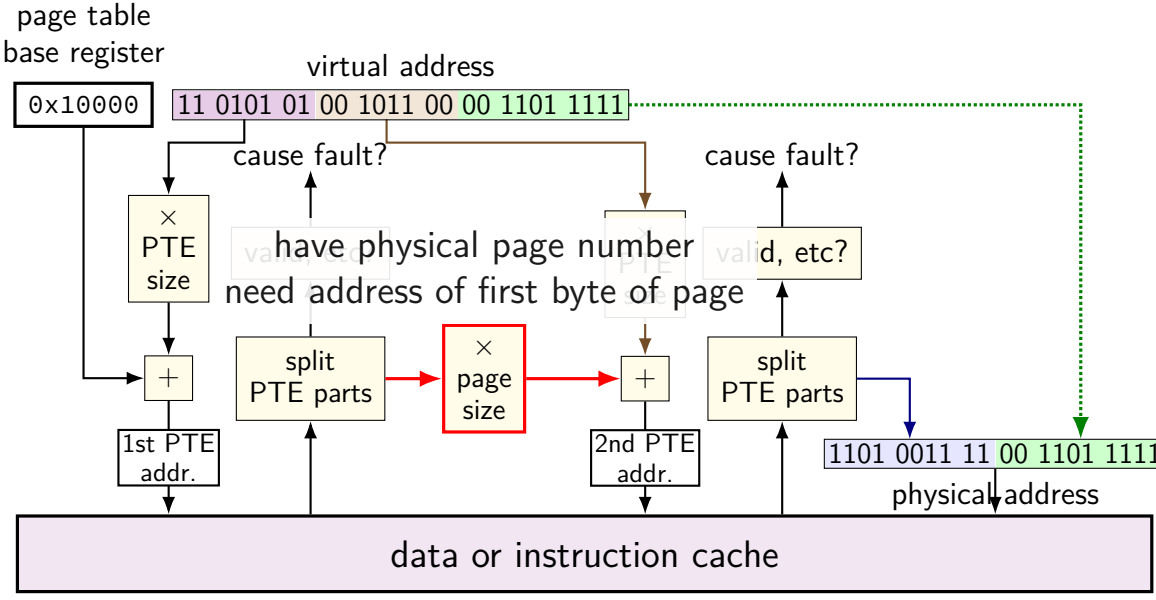
two-level page table lookup



two-level page table lookup



two-level page table lookup



two-level page table lookup