Base and limit registers

CPU

Logical address

Limit register

Relocation register

Physical address

Memory

<

yes

no

trap: addressing error

>
Segmented addressing

- CPU
- Segment table
- Memory
- Trap: addressing error

Process:
1. CPU sends segment and data information to the segment table.
2. If the address is within the segment limit and base, it proceeds to the next step.
3. If the address is out of bounds, a trap occurs indicating an addressing error.
4. For valid addresses, the physical address is calculated by adding the base address to the segment address.

Diagram:
- CPU sends data (s, d) to the segment table.
- Check if the address is less than the segment limit and base.
- If yes, calculate the physical address.
- If no, trap: addressing error.

Physical address calculation:
- Base address + Segment address

Segmented addressing involves dividing memory into segments, allowing for more efficient management and protection of memory resources.
Hashed Page Table

logical address

hash function

hash table

physical memory

physical address

r d

p d

q s

p r

• • •
Inverted Page Table Architecture
P6 protection bits

Page base address: 20 most significant bits of physical page address (forces pages to be 4 KB aligned)

Avail: available for system programmers

G: global page (don’t evict from TLB on task switch)

D: dirty (set by MMU on writes)

A: accessed (set by MMU on reads and writes)

CD: cache disabled or enabled

WT: write-through or write-back cache policy for this page

U/S: user/supervisor

R/W: read/write

P: page is present in physical memory (1) or not (0)

P bit can be used to trap on any access (read or write)
MIPS R4x00 synonyms

- ASID-tagged, on-chip L1 VIPT cache
  - 16kB cache, 32B lines, 2-way set associative
  - 4kB (base) page size
  - Set size = 16kB/2 = 8kB > page size
  - Overlap of tag & index bits, but from different addresses!

- Remember, location of data in cache determined by index
  - Tag only confirms whether it’s a hit
  - Synonym problem iff $VA_{12} \neq VA'_{12}$
  - Similar issues on other processors, e.g. ARM11 (set size 16kB, page size 4kB)
Cache bombs

- Unmap page with dirty cache line
- Re-use (remap) frame to a different page (in same or different AS)
- Write to a new page
  - Without mismatch, new write overwrites old (hits same cache line)
  - With mismatch, order can be reversed: “cache bomb”
Aliasing

- Page aliased in different address spaces
  - AS1: $VA_{12} = 1$, AS2: $VA_{12} = 0$
- One alias gets modified
  - In a write-back cache, other alias sees stale data
  - Lost-update problem