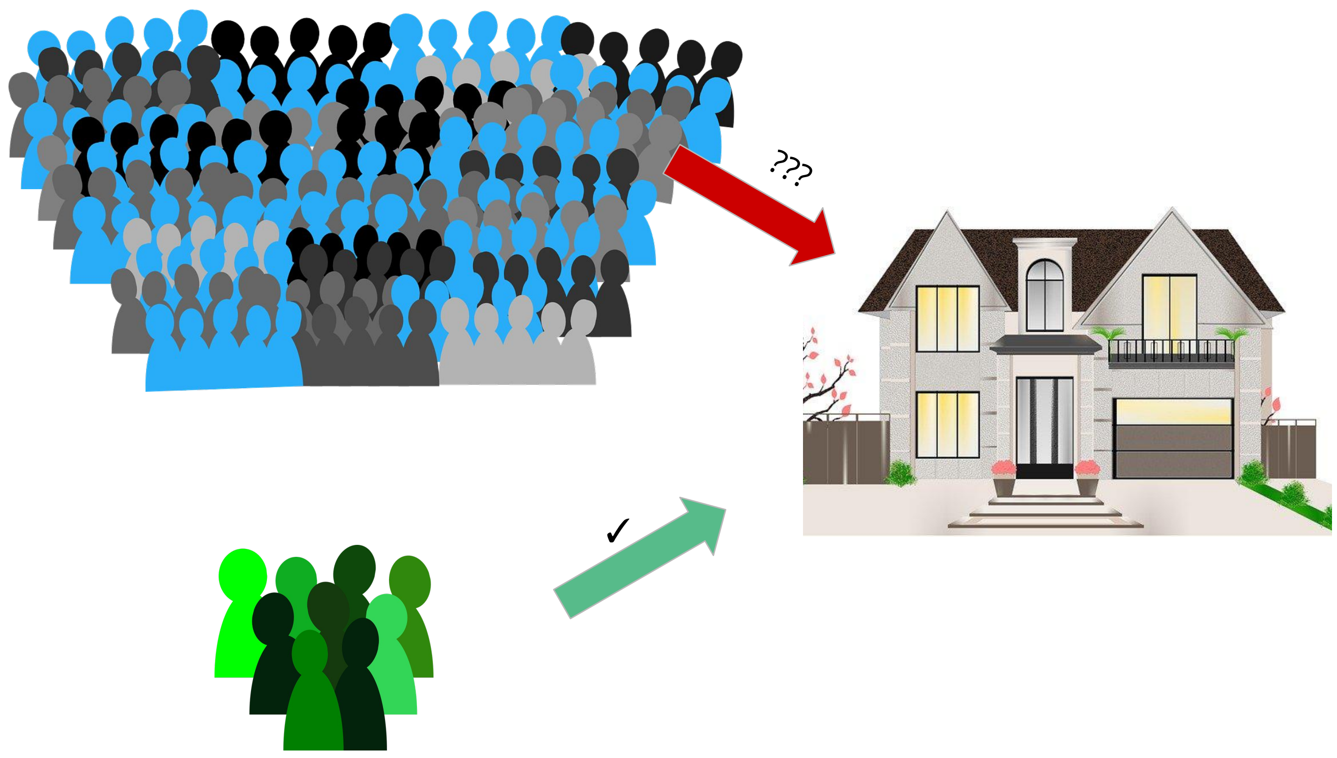
Non-Contiguous Allocation

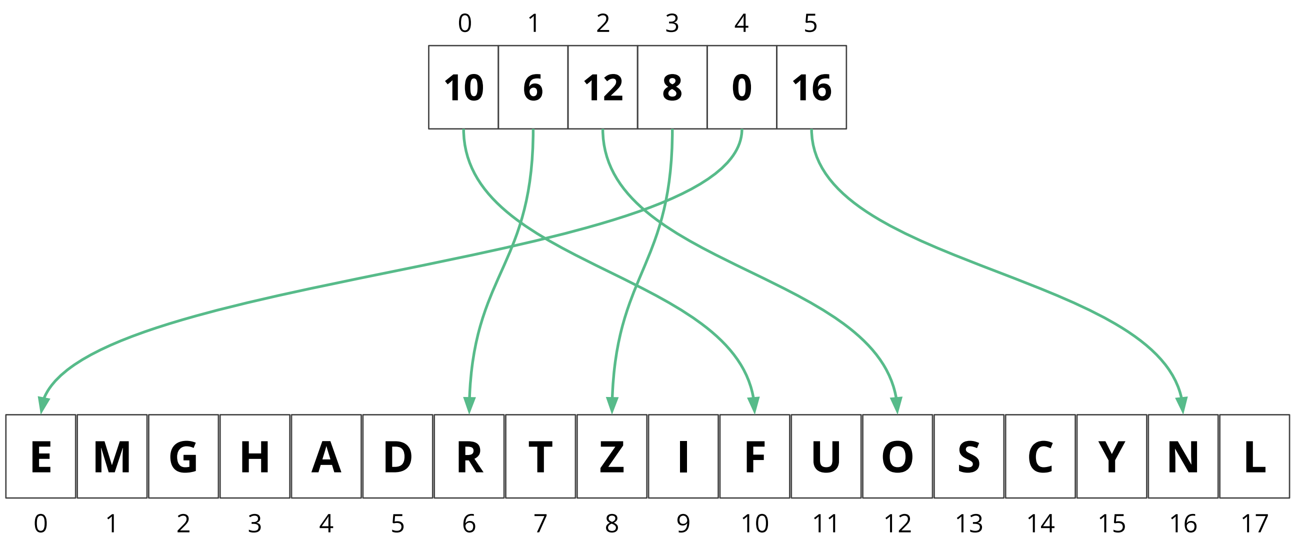
Hosting a party for N people



Fact #4a: code and data of a process may be split into "units" placed **non-contiguous**ly from each other

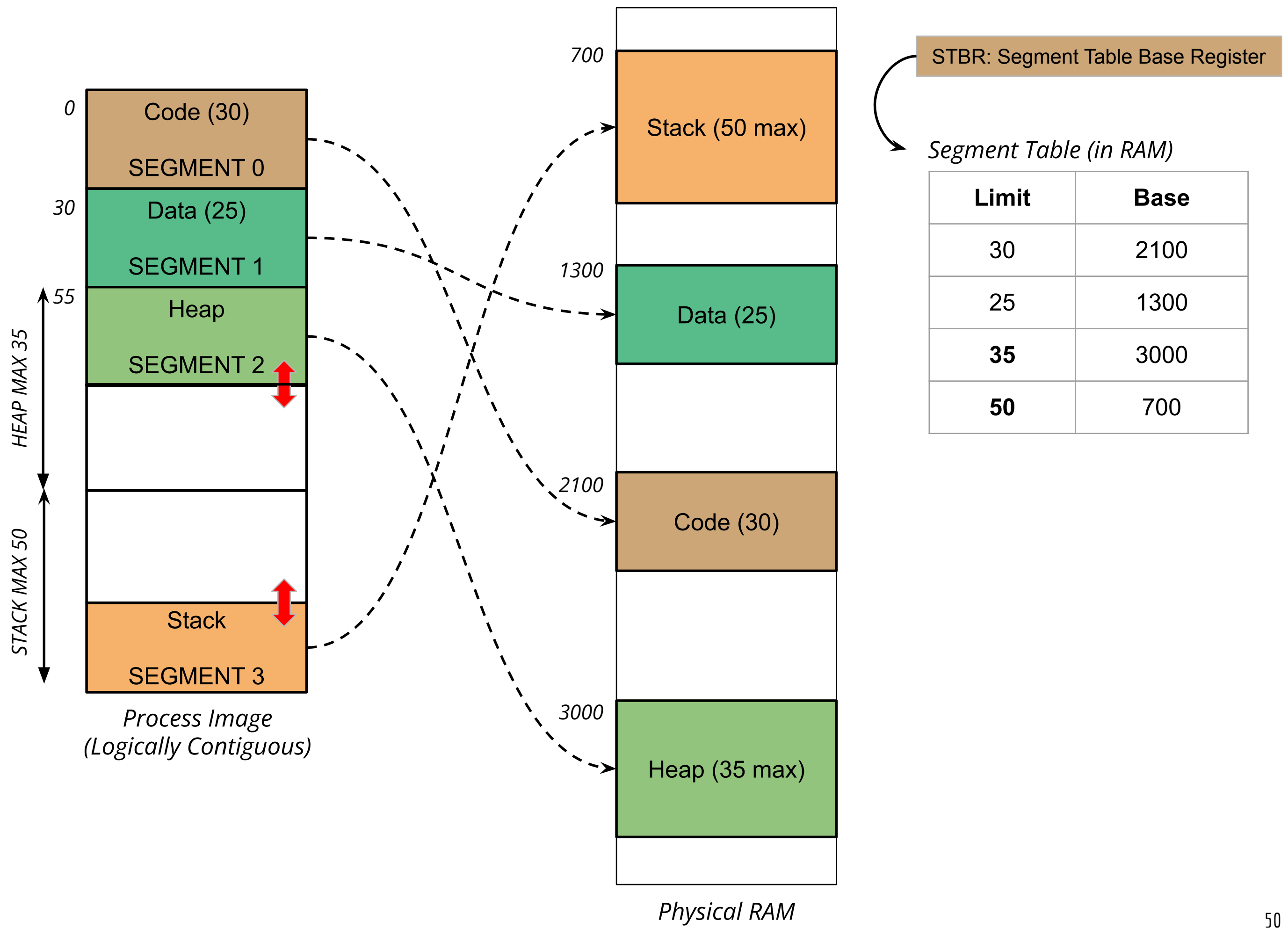
Fact #4b: each unit itself is contiguous

Word Puzzle

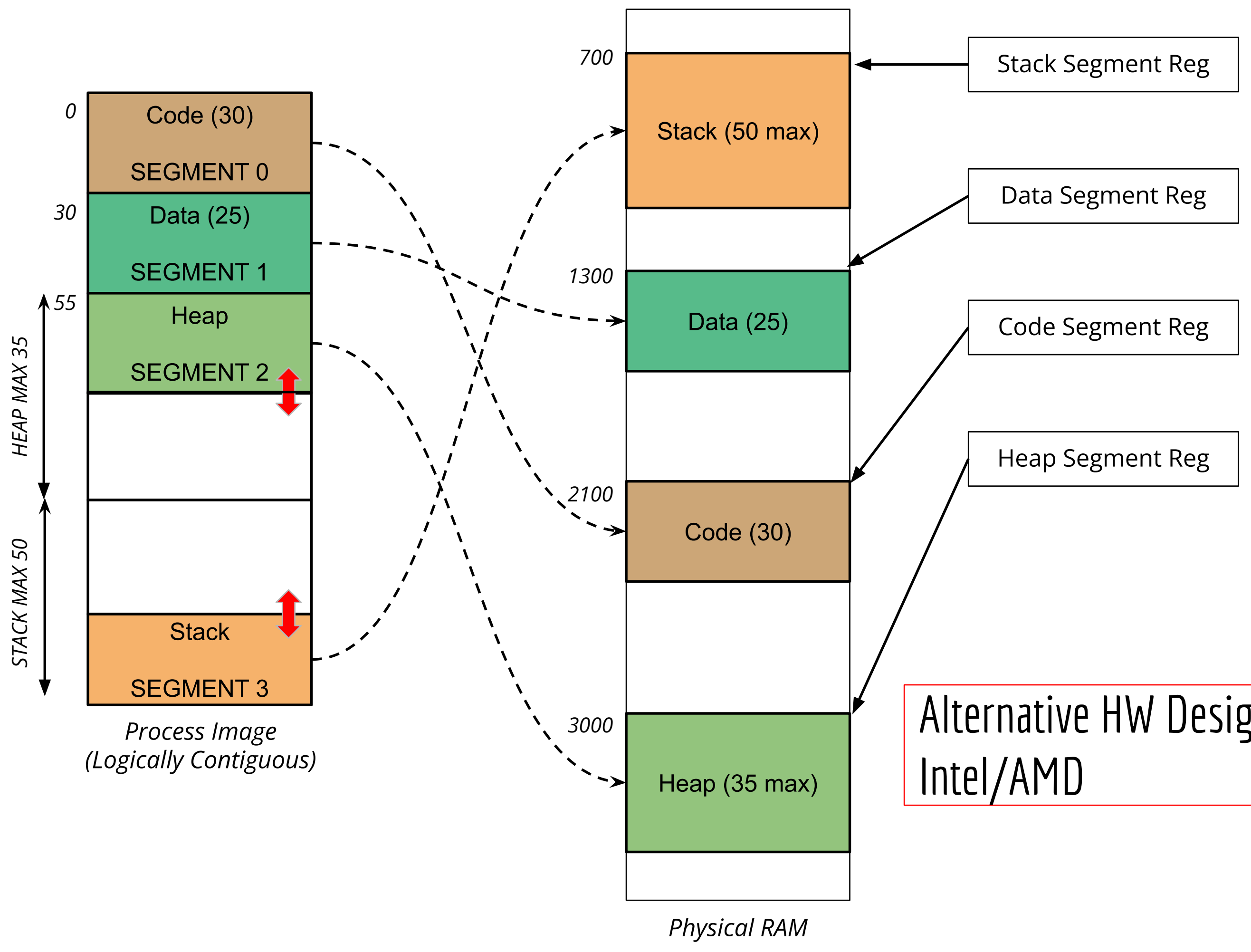


“Non-Contiguous” Memory Allocation

- Segmentation
  - Split a process memory into several (**non-uniform size**) segments
  - Each segment corresponds to a logical unit (typically created by the compiler)
    - Code [section/segment]
    - Read-only data [section/segment] and R/W data [section/segment]
    - Stack [section/segment]
    - Heap [section/segment]
    - Global data, uninitialized data, ...
  - Each segment itself is **contiguous**, but the segments themselves may NOT be
- Paging
  - Split a process memory into (**uniform size**) non-contiguous pages
- **Can’t use just ONE pair of Base Register & Limit Register anymore. Why?**

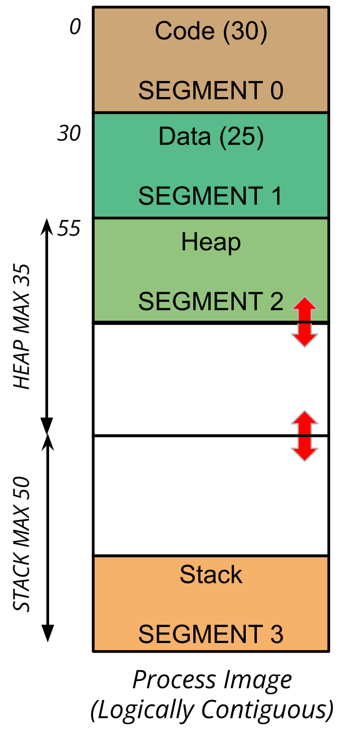


Limit	Base
30	2100
25	1300
35	3000
50	700



Alternative HW Design
Intel/AMD

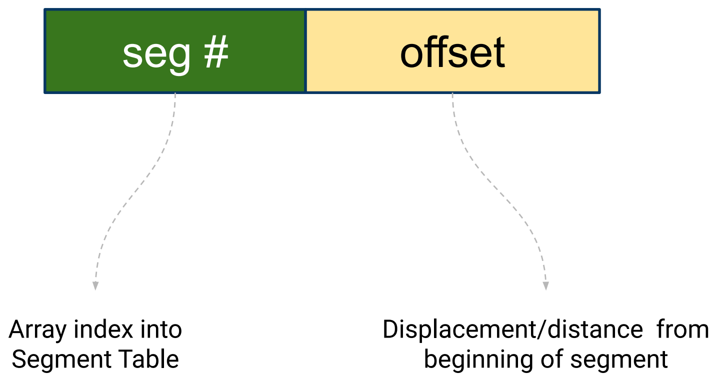
Compiler Redesign (to support segmentation)



Code at logical address 27 ⇒ Segment #0 Offset 27

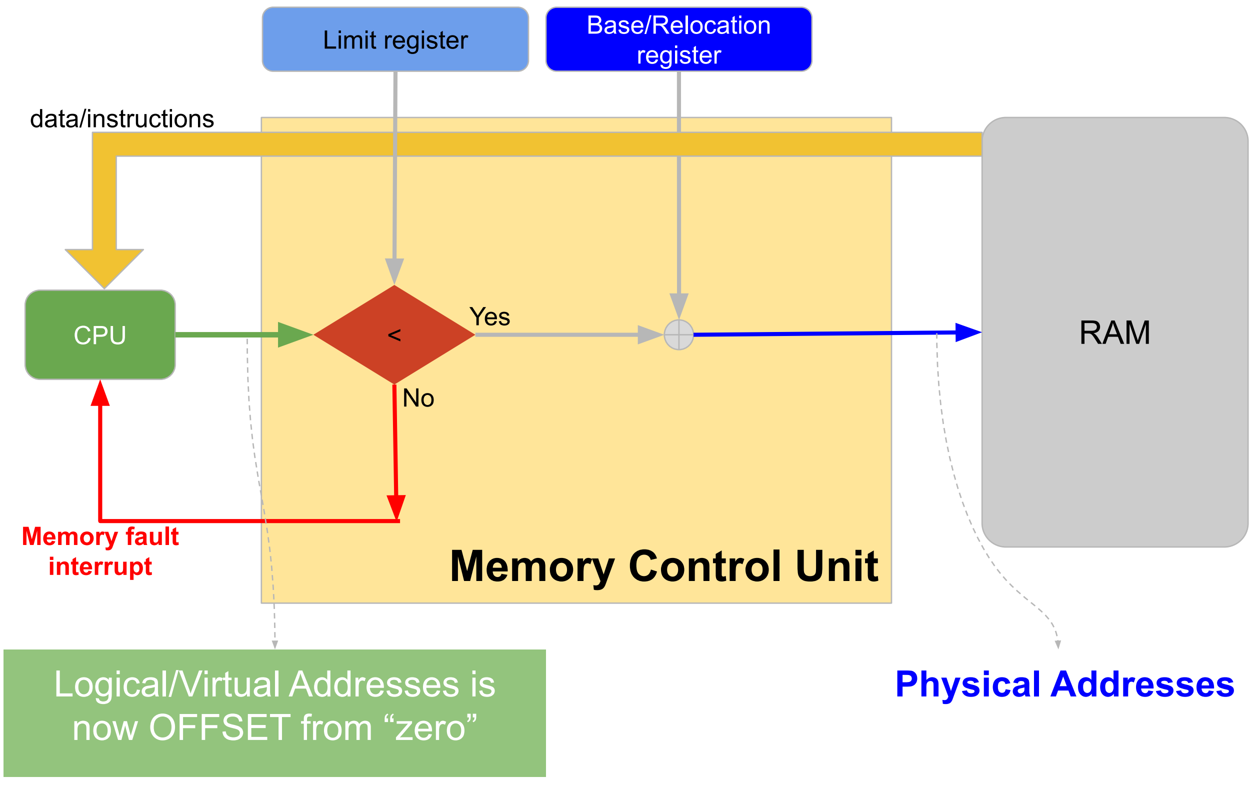
Data at Logical Address 35 ⇒ Segment #1 Offset 5

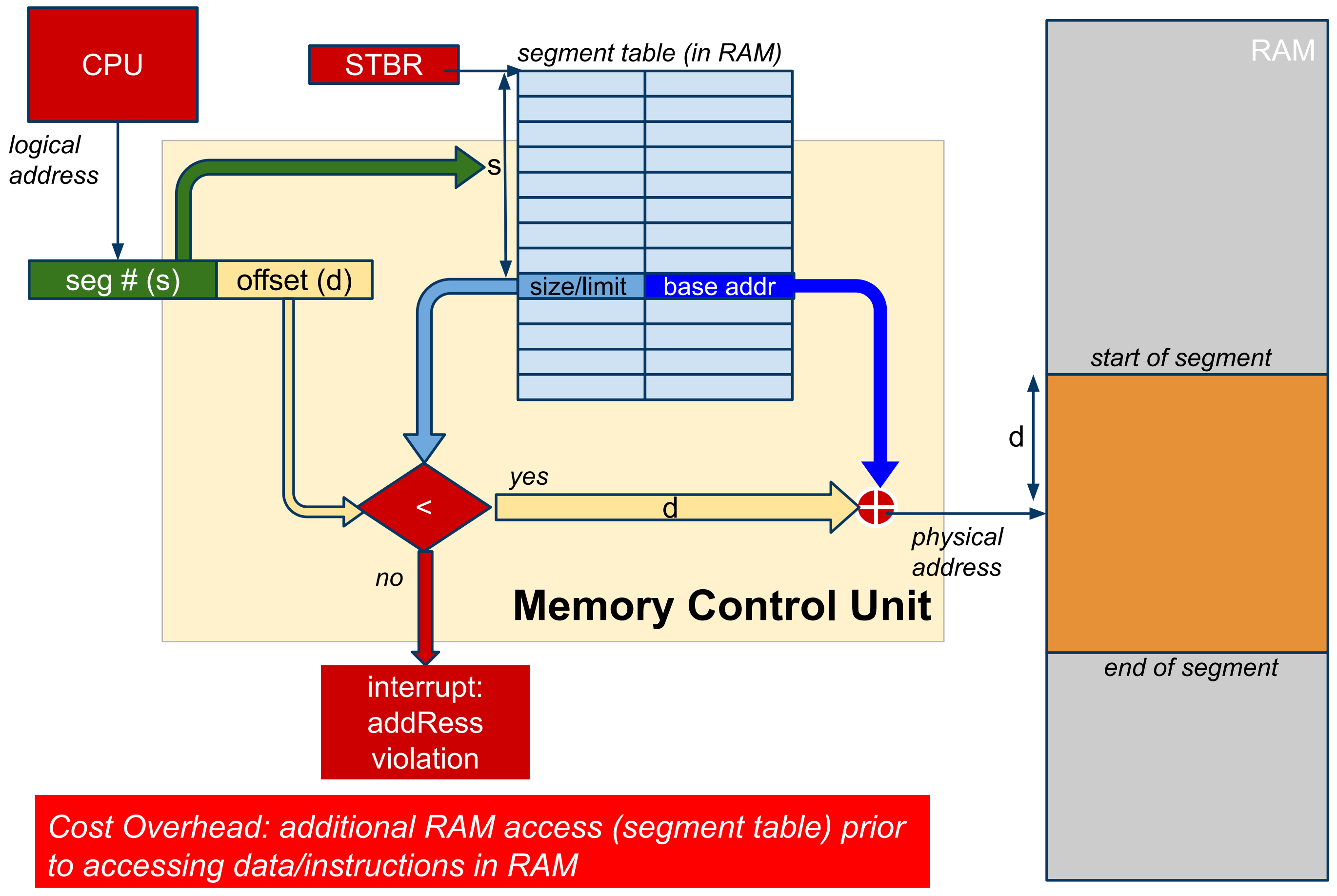
Heap item at logical addr 67 ⇒ Segment #2 Offset 12



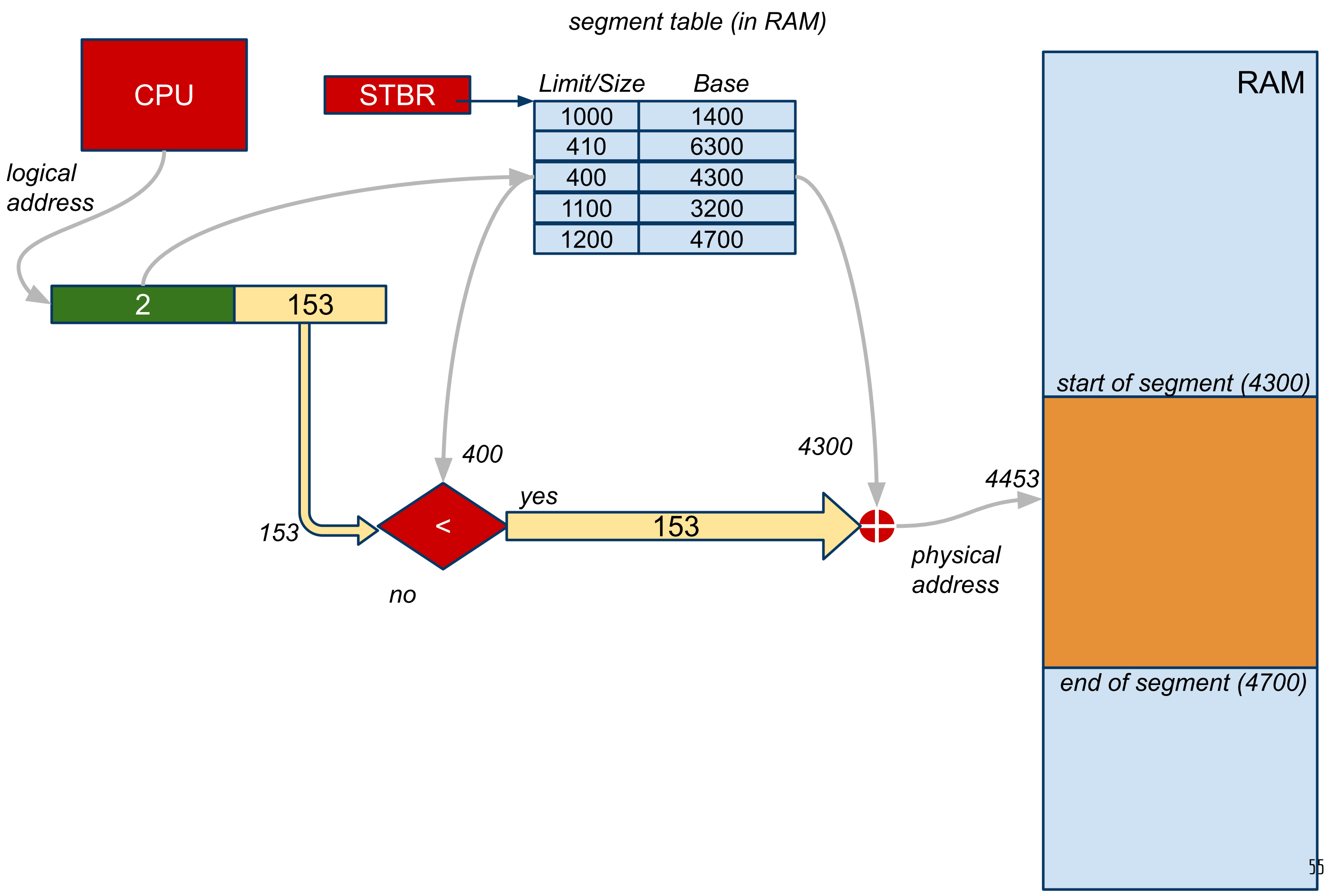
seg #
offset
Array index into Segment Table
Displacement/distance from beginning of segment

Base & Limit Regs (0-based Logical Addr.)



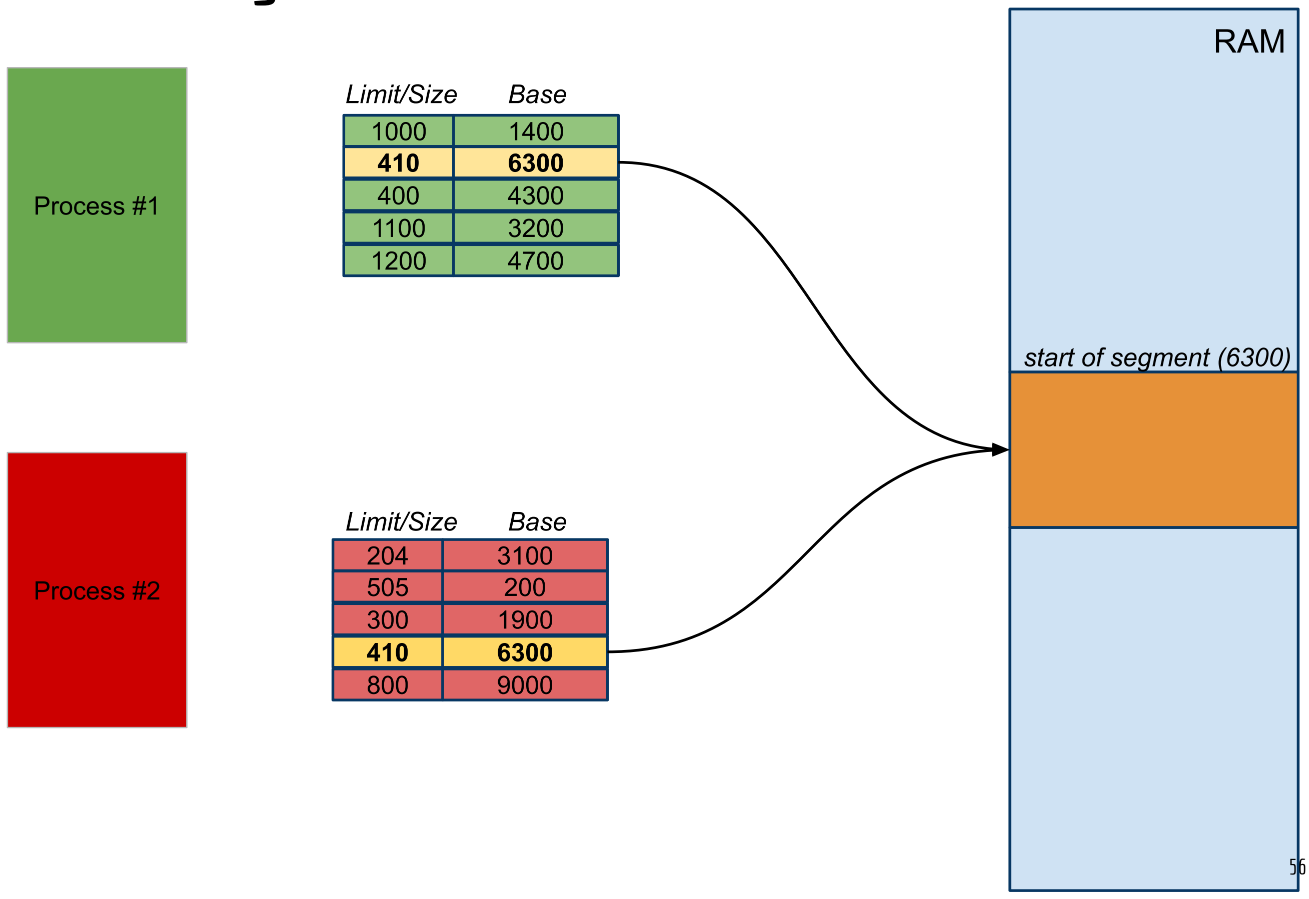


Cost Overhead: *additional RAM access (segment table) prior to accessing data/instructions in RAM*



Limit/Size	Base
1000	1400
410	6300
400	4300
1100	3200
1200	4700

Shared Segment(s)



Limit/Size	Base
1000	1400
410	**6300**
400	4300
1100	3200
1200	4700

Limit/Size	Base
204	3100
505	200
300	1900
410	**6300**
800	9000

Segmentation: Memory Address

- Logical address generated the CPU has two parts
  - Segment number (used for indexing the segment table)
  - Offset within the segment
- The OS manages segment tables (**one table per process**). Each segment table entry consists of:
  - Size of the segment (mimics the Limit Register)
  - Start location of the segment (mimics the Base/Relocation Register)
- The address of the segment table is kept in Segment Table Base Register (STBR)
  - STBR must be saved/restored on context-switch

Case Study: Intel x86 CPUs

Intel x86 (32-bit) Segment Registers

Register	Description	Usage
CS	Code Segment Register	Access the code
DS	Data Segment Register	Access the data
SS	Stack Segment Register	Access the stack
ES	Extra Segment Register	String copy/compare operations

On recent Intel CPUs with 64-bit architecture these registers are not used (set to 0) when the CPU is running in 64-bit mode!

Quiz: Segmentation & Segment Table