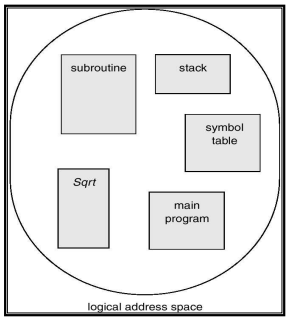
Segmentation

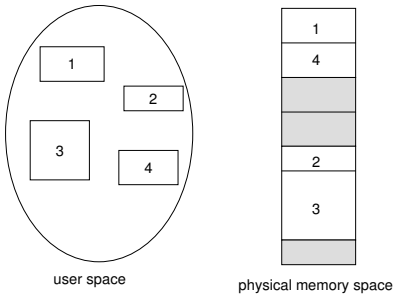
Segmentation

- Memory-management scheme that supports user view of memory.
- A program is a collection of segments. A segment is a logical unit such as:
  - main program
  - procedure
  - function
  - method
  - object
  - local variables, global variables
  - common block
  - stack
  - symbol table, arrays

User's View of a Program



Logical View of Segmentation



Segmentation

- A straightforward solution to our compiler problem is to give each are an independent address space, called a *segment*.
- Each segment consists of linear addresses from zero up to a maximum.
- Different segments can have different lengths.
- A segment's length may even change dynamically during execution (e.g., a segment for a stack).
- Each virtual address has two parts:

segment-number	offset

Segmentation = Paging with variable size pages

Use of Segments

Suppose each procedure is a separate segment.

1. If procedure in segment n is recompiled, no other procedures need be changed.

   In contrast, a 1-dimensional address space requires a linker to layout all procedures compactly, thus affecting many procedure's entry point addresses.
2. A shared library can put each sharable unit in a different segment.

   A paged system essentially simulates segmentation (by putting library elements on page boundaries) to permit shared libraries.

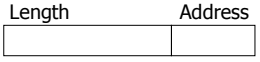
Segmentation Memory Management

- Segments are brought in on demand
- Uses variable size partitioning:
  - Contiguous allocation of each segment
  - Each segment must occupy contiguous
  - locations, but segments may be scattered throughout memory.
  - Use first fit, etc.

Segmentation: Implementation

- Divide each process into logical segments (procedures, arrays, etc.)
- Logical breakdown gives an intuitive structure to main memory
- Managing segments:

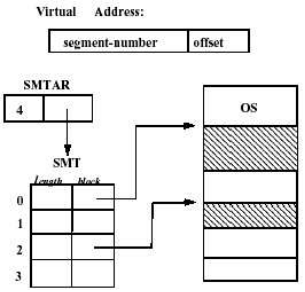
Segment Map Table (SMT)



1 SMT/Job

1 entry/segment

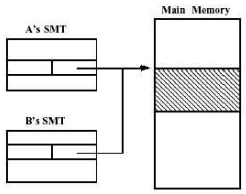
Segment addressing



- Each offset is checked against the segment length to ensure legal access

Sharing segments

- Because segmentation is based on a logical ordering instead of a physical one, *sharing and protection are easier to implement*
- Share SQRT function, instead of sharing particular pages



Segmentation: Pros and Cons

Advantages:

1. Multiple VA spaces
2. Whole program need not be simultaneously in memory
3. Sharing and protection are easier than any memory management method discussed so far

Segmentation: Pros and Cons

Disadvantages:

1. VA to PA translation requires table lookup, increasing memory cycle time
2. Entire segment must be in memory simultaneously -- a problem with big segments!
3. SMT needs one more field than PMT: for segment length
4. Variable size storage allocation requires compaction
5. External fragmentation

Segmentation with Demand Paging

- To handle the problem of large segments, divide each logical segment into fixed size pages

Virtual Address

segment	page	offset

Example

1	2	100

offset 100 from page 2 of segment 1

Managing Segments and Pages

1. SMTAR (analogous to PMTAR)
   - Length (# of segments)
   - Address of SMT
2. SMT
   - Length (# of pages) for each segment
   - Address of PMT for each segment
   - Residence bit
   - 1 SMT / job
   - 1 entry / segment
3. PMT
   - Status (S, M, IT) of each page
   - Block # for each page
   - 1 PMT / segment
   - 1 entry / page

Pros/Cons of Segmentation with paging

Advantages:

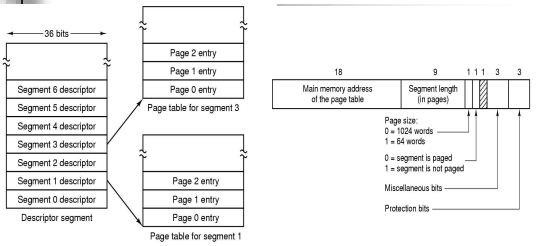
1. Can page unused PMTs
2. Multiple virtual address spaces simplify programming
3. Easy memory allocation
4. Easy sharing and protection
5. Only part of segment need be in memory at once

Pros/Cons of Segmentation with paging

Disadvantages:

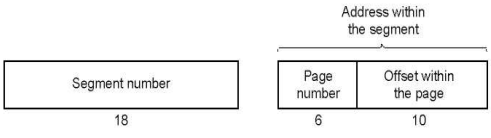
1. Even more overhead than segmentation for virtual to physical address translation
2. Internal fragmentation
3. Storage space for both SMTs and PMTs

Segmentation with Paging: MULTICS (1)



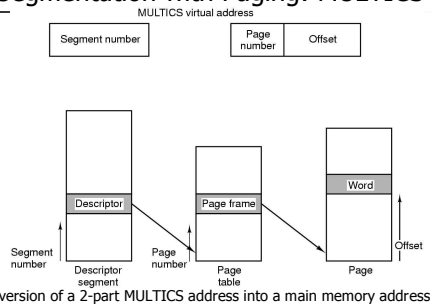
- Descriptor segment points to page tables
- Segment descriptor – numbers are field lengths

Segmentation with Paging: MULTICS (2)



A 34-bit MULTICS virtual address

Segmentation with Paging: MULTICS (3)



Conversion of a 2-part MULTICS address into a main memory address

A 34-bit MULTICS virtual address

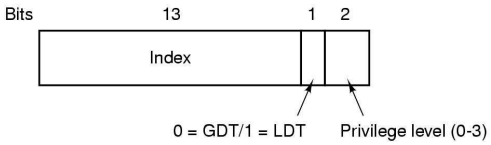
Segmentation with Paging: MULTICS (4)

Segment number	Virtual page	Page frame	Protection	Age	Is this entry used?
4	1	7	Read/write	13	1
6	0	2	Read only	10	1
12	3	1	Read/write	2	1
					0
2	1	0	Execute only	7	1
2	2	12	Execute only	9	1

Comparison field

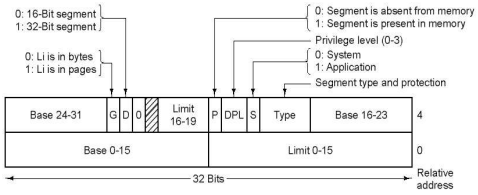
- Simplified version of the MULTICS TLB
- Existence of 2 page sizes makes actual TLB more complicated

Segmentation with Paging: Pentium (1)



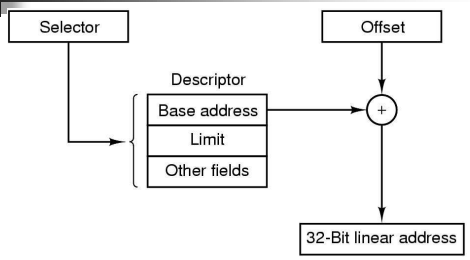
A Pentium selector

Segmentation with Paging: Pentium (2)



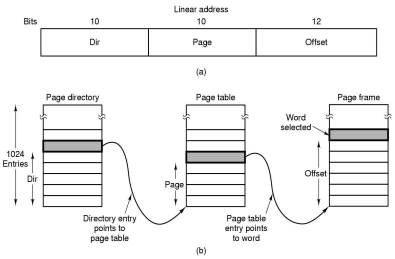
- Pentium code segment descriptor
- Data segments differ slightly

Segmentation with Paging: Pentium (3)



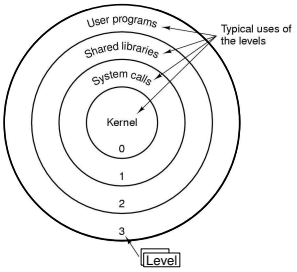
Conversion of a (selector, offset) pair to a linear address

Segmentation with Paging: Pentium (4)



Mapping of a linear address onto a physical address

Segmentation with Paging: Pentium (5)



Protection on the Pentium