Approaches to Noncontiguous Memory Allocation

- Paging
  - Process consists of fixed-size components called *pages*
    - The page size is defined by hardware
  - Eliminates external fragmentation

- Segmentation
  - Programmer identifies logical entities in a program; each is called a *segment*
  - Facilitates sharing of code, data, and program modules between processes
  - External fragmentation can arise

- Hybrid approach: *segmentation with paging*
  - Facilitates sharing of code, data, and program modules between processes
  - Avoids external fragmentation

Memory Protection

- Memory areas allocated to a program have to be protected against interference from other programs
  - MMU performs this through a bounds check
    - While performing address translation for a logical address \((comp, \text{byte})\), MMU checks if \(comp\) actually exists in program and whether \text{byte} exists in \(comp\)
      - Protection violation interrupt raised if check fails
  - Bounds check can be simplified in paging
    - \text{byte}, cannot exceed size of a page
Paging

- In the logical view, the address space of a process consists of a linear arrangement of pages
- Each page has \( s \) bytes in it, where \( s \) is a power of 2
  - The value of \( s \) is specified in the architecture of the computer system
- MMU decomposes a logical address into the pair \((p_i, b_i)\), where \( p_i (\geq 0) \) is the page number and \( b_i (0 \leq b_i < s) \) is the byte number within page \( p_i \)

![Figure 11.19](image1.png) Logical view of processes in paging.

Paging (continued)

- Memory is divided into areas called page frames
- A page frame is the same size as a page
- Kernel constructs a page table for each process

![Figure 11.20](image2.png) Physical organization in paging.
Paging (continued)

- Notation used to describe address translation:
  - $s$: Size of a page
  - $l_l$: Length of a logical address (i.e., number of bits in it)
  - $l_p$: Length of a physical address
  - $n_b$: Number of bits used to represent the byte number in a logical address
  - $n_p$: Number of bits used to represent the page number in a logical address
  - $n_f$: Number of bits used to represent frame number in a physical address
- The size of a page, $s$, is a power of 2
  - $n_b$ is chosen such that $s = 2^{n_b}$

\[
\begin{array}{c|c|c|c}
\text{logical address} & \text{start address of frame $q_i$} & \text{physical address} \\
\hline
p_i & b_l & q_i & b_f \\
\hline
l_l & n_f & l_p & n_f \\
\end{array}
\]

Page $p_i$ is allocated page frame $q_i$

Example: Address Translation in Paging

- 32-bit logical addresses
- Page size of 4 KB
  - 12 bits are adequate to address the bytes in a page
    - $2^{12} = 4$KB
- For a memory size of 256 MB, $l_p = 28$
- If page 130 exists in page frame 48,
  - $p_i = 130$, and $q_i = 48$
  - If $b_i = 600$, the logical and physical addresses are:

\[
\begin{array}{c|c|c|c}
\text{Logical address} & \text{Physical address} \\
\hline
0 \ldots 010000010 & 001001011000 \\
\hline
0 \ldots 00110000 & 001001011000 \\
\end{array}
\]
## Summary

- Compiler assumes a specific memory area to be available to program and generates *object module*
- *Linker* performs *relocation* of a program, and performs *linking* to connect the program with library functions
- CPU has a *relocation register* to facilitate relocation
- Memory allocation can be: static or dynamic
  - Code and static data of program are statically allocated
  - Stack and heap of program are dynamically allocated
- Allocation/deallocation of memory can lead to fragmentation: internal or external
  - *Noncontiguous allocation* reduces external fragmentation
    - Requires use of the memory management unit (MMU) of CPU