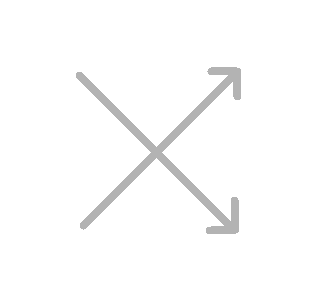
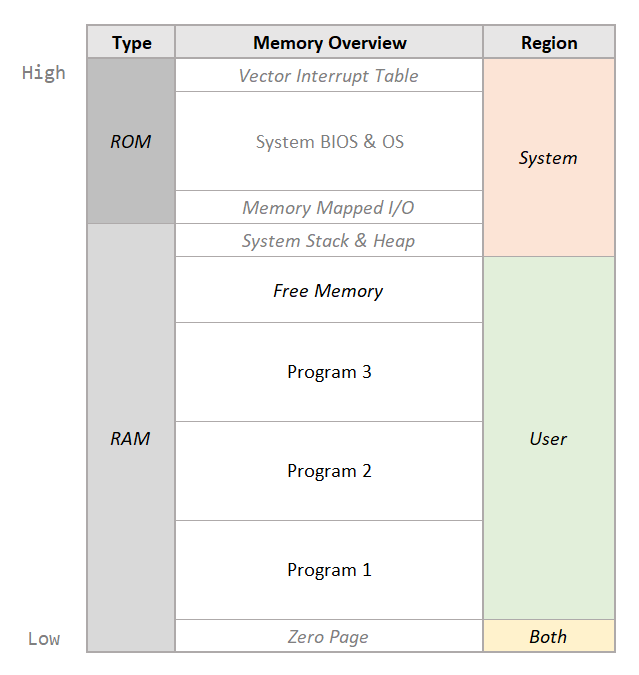
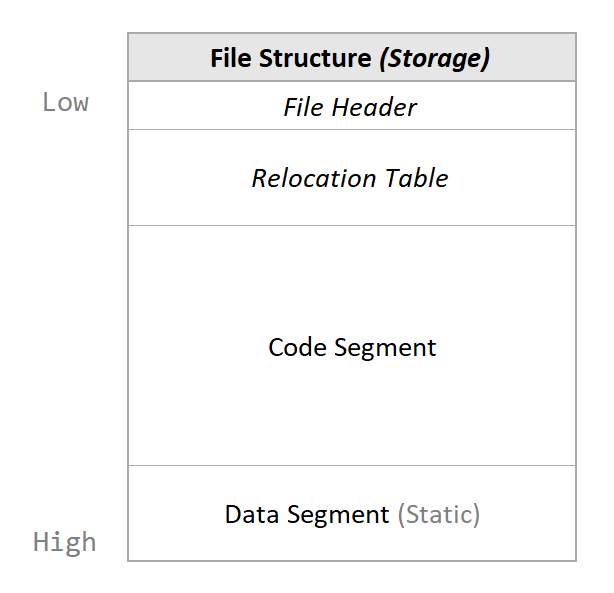
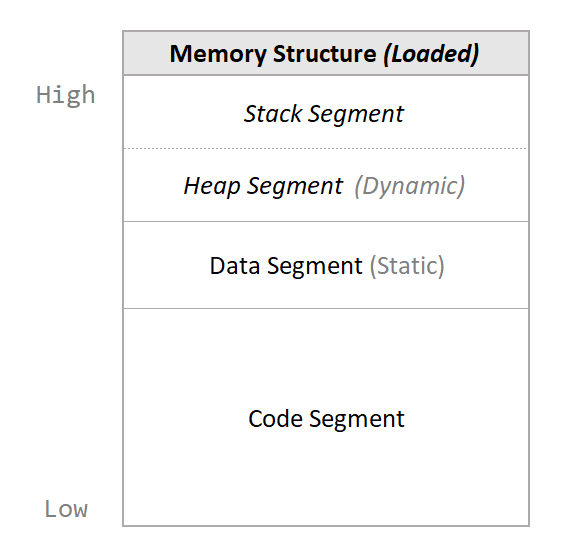
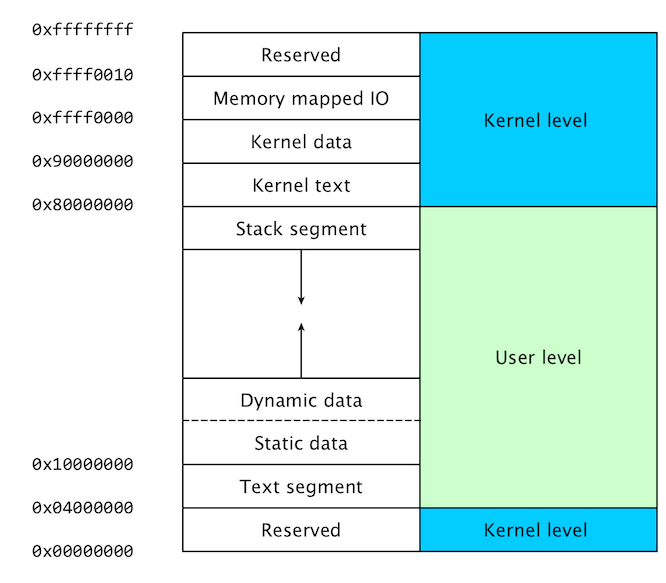
******Relocatable Binary Format**

1. **File Header**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entry** | **Location** | **Bytes** | **Description** |
| Magic Identifier | 00 – 01 | 2 | Denotes relocatable binary file. *(“RB” in ASCII.)* |
| Format Version | 02 – 03 | 2 | Version number of the file format. *(01 00h = v1.0)* |
| Binary Type | 04 – 05 | 2 | Properties of this binary file.  *Executable, relocatable, position-independent, single…* |
| Binary Target | 06 – 07 | 2 | Architecture and variant (ABI, wordsize? endianness?).  *(01XXh = Abacus 2000)* |
| Origin Address | 08 – 09 | 2 | Address to load code into memory at. *(Default 0100h.)*  *Used to calculate whether program can fit in memory.* |
| Header Size | 0A – 0B | 2 | Total size of this binary header.  *Used to calculate when to stop reading header.* |
| Relocation Table Size | 0C – 0D | 2 | Size of relocation table. |
| Code Size | 0E – 0F | 2 | Size of program code segment *(sans header)*.  *Used to calculate if program can fit in memory.* |
| Data Size | 10 – 11 | 2 | Size of static data segment.  *Used to calculate if program can fit in memory.* |
| Heap Size | 12 – 13 | 2 | Desired size of dynamic data/heap segment.  *Used to reserve space for dynamic data. (00h = max.)* |
| Stack Size | 14 – 15 | 2 | Desired size of stack segment.  *Used to reserve space for stack. (00h = max.)* |

1. **Binary Type**

Each bit represents a true or false value referring to each of the following properties…

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Executable Binary | Position Independent | Relocatable Code | Library Only | Selfish Program | Single Process |  |  |
|  |  |  |  |  | Endianness?  e.g. Little | Word Length?  e.g. 16-bit | |

* Executable Binary: Is this binary executable code?
* Position Independent: Can the code be loaded anywhere in memory *without* modification?
* Relocatable Code: Can the code be loaded anywhere in memory *with* modification? I.e. Can the absolute references to addresses be relocated? *(Requires address relocation table.)*
* Library Only: Is this binary a shared library only? I.e. Can it *not* be executed independently?
* Selfish Program: Must this program be the *only* program loaded in this memory (segment)?
* Single Process: Must only *one instance* of this program ever be loaded at any one time?
* Word Length: 16-bit or 32-bit (or 64-bit)?
* Endianness: Are values encoded in big or little endian?

1. **Binary Target**

|  |  |
| --- | --- |
| Target Architecture *(e.g. Abacus 2000)* | ABI/OS/BIOS dependencies *(e.g. ASMOS 1.0)*  Word Size? Endianness? |

1. **Relocation Table**

A table containing the addresses of literal values in the machine code that refer to memory locations. *E.g. In absolute JUMP or LOAD/STORE instructions.*

If program is being loaded at a memory location *other than* the specified Origin Address, the loader must modify each of the values pointed to in the Table so that they point to the correct addresses.

Every value pointed to by the Table with be a 16-bit unsigned integer *(because 8-bit literals will always either be relative or pointing to the Zero Page)*.

1. **Code Segment**

Machine code. A continuous stream of bytes that will be copied directly into working memory, starting at the origin.

If file is marked as relocatable, each address listed in the Relocation Table will be updated after the program is loaded into memory.

1. **Data Segment**

Initialized variables, arrays, etc. To be loaded into memory directly following the program code. Constants to be loaded first.

*Heap segment and stack segment not included in file because that data is dynamically allocated.*