Using Overlays

Overlay (shared library) Overview

Methods for reading and executing a program from a CD

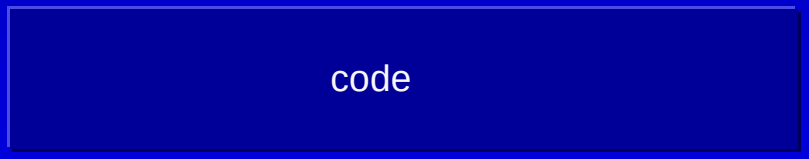
- Read an EXE file using CD function and Exec()
- Use LoadExec()
- Using Overlay(s)

Execution file Format

.EXE file

- Launch from CD-ROM
- Length in multiples of 2048

.BIN file

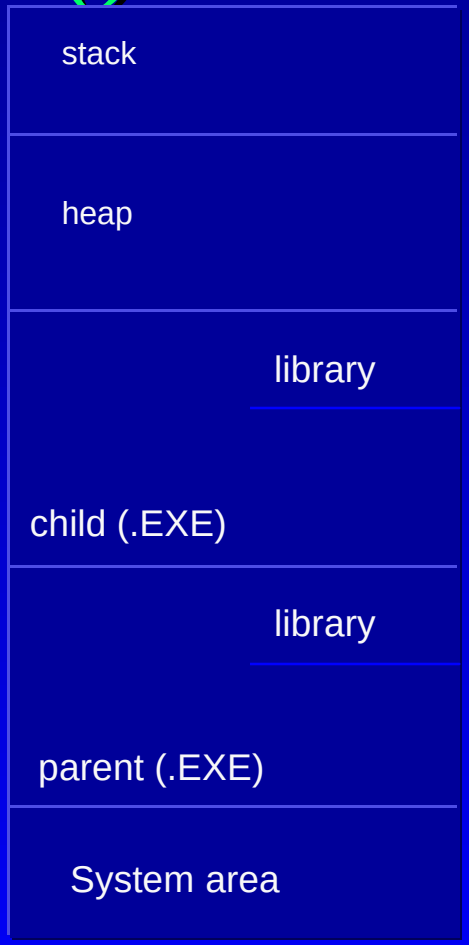


- Launch from CD-ROM
- Overlay

Read in an EXE file and then Exec()

- Advantage:
  - Can be developed independently and run as a child program
- Disadvantages:
  - When control is transferred it is necessary to stop all interrupts.
  - Two copies of the Libraries reside in RAM

Read in an EXE file and then Exec()



Initial value of parent stack pointer

Initial value of child stack pointer

Load address of child

Load address of parent

Method using LoadExec()

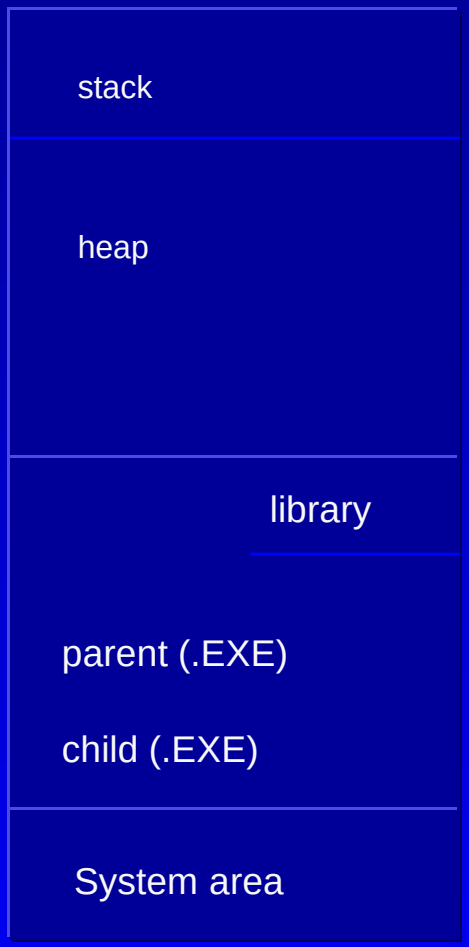
- Advantages:
  - Can be developed independently and run as a child program
  - Can load over the parent
- Disadvantages:
  - When control is transferred it is necessary to stop all interrupts
  - _96_init() must be called
  - and the biggie...

Method using LoadExec()

- If LoadExec() fails:

There is *NO* Recovery

Method using LoadExec()



Initial value of parent or child stack pointer

Load address of parent or child

Method using Overlay

- Overlays are segments of code which can be manipulated so they “take turns” occupying the same memory location
- Possible examples:
  - animations, levels / scenarios, CD functions, memory card, menus

Method using Overlay

- Advantages:
  - Library is shared between parent and child
  - Not necessary to stop interrupts
  - A separate process can be executed while the child is loading
  - Can utilize function parameters

Method using Overlay

- Disadvantages:
  - The make operation is more complex
  - Libraries must be completely linked into the parent program
    - Size of the parent is larger

Method using overlay

Initial value of stack pointer
(shared between parent and child)

Load address of child

Load address of parent

Overlay Programing

Overlay programming

- Makefile
- Link file
- Address file
- Parent/loader

Sample Program

- Parent: Menu
  - Select the overlay and load it.
- Children:
  - balls(Sample sprite display)
  - rcube(Sample 3D display)
  - anim:(Sample movie)

Makefile

- Using the link file with psylink

  psylink /c /m @menu.lnk,menu.cpe, menu.sym, menu.map

- Child compile options

  -G0 -Wa,s[name]

  ccpsx -O3 -G0 -c -Wa,sballs balls.c

Link file

- General link file description
  - org setting
  - group setting
  - section setting
  - include file
  - inclib file
  - pc initial value setting

Link file

- org setting
  - The target machine code location is made known to the assembler
    - Example: **org $80010000**

Link file

- group setting
  - A group is a collection of sections.
  - You can set attributes of each group
    - bss : uninitialized global data
    - org(address) : org setting address
    - file(file) : output binary as a file
    - over(group) : overlay to group

Link file

- group setting example

```
text        group  org($80010000)
bss         group  bss
balls       group  file("balls.bin")
rcube       group  over(balls),file("rcube.bin")
anim        group  over(balls),file(“anim.bin”)
```

Link file

- section setting
  - Allocate each section to a group
  - Normally, the compiler creates a section in the six parts below
    - .rdata read only data
    - .text code
    - .data initialized data
    - .sdata initialized data (small)
    - .sbss uninitialized data (small)
    - .bss uninitialized data

Link file

- Section setting example

```
section    .rdata,text
section    .text,text
section    .data,text
section    .sdata,text
section    .sbss,bss
section    .bss,bss
```

Link file

- include file
  - Describes the file that is linked
    - Example: include main.obj
- inclib file
  - Describes the library file
    - Example: inclib "c:\psx\lib\libapi.lib”

Link file

- pc initial value setting
  - Normally set to __SN_ENTRY_POINT
    - Example:

      regs pc=_ _SN_ENTRY_POINT

Address file

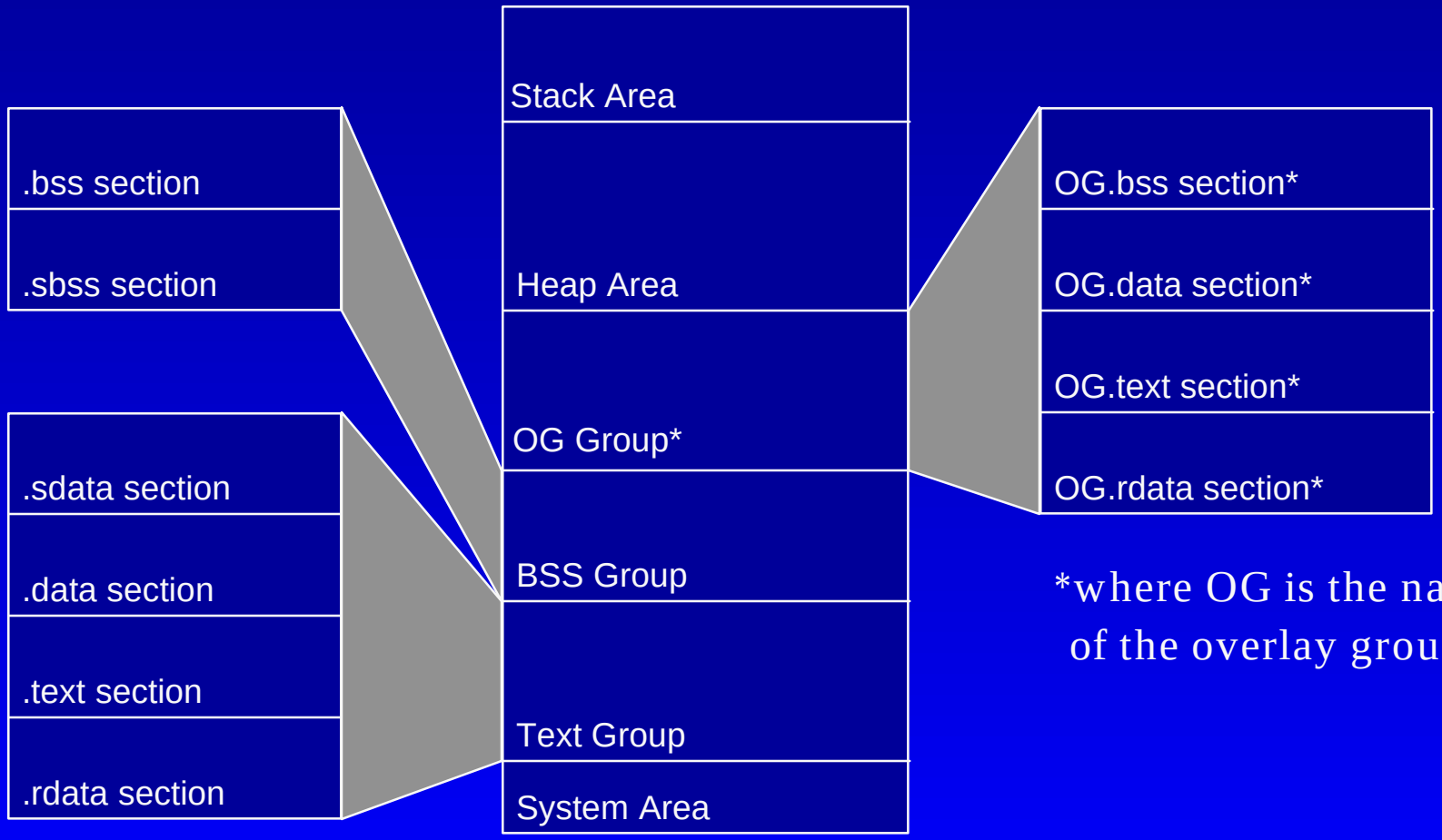
- Set load address of child program

```
                opt         c+
balls           group
                xdef        LoadAddress
                section     .rdata
LoadAddress     dw          group(balls)
```

Parent loader description

- As needed, read the BIN file to the LoadAddress and call the module as a function

Typical Memory Map

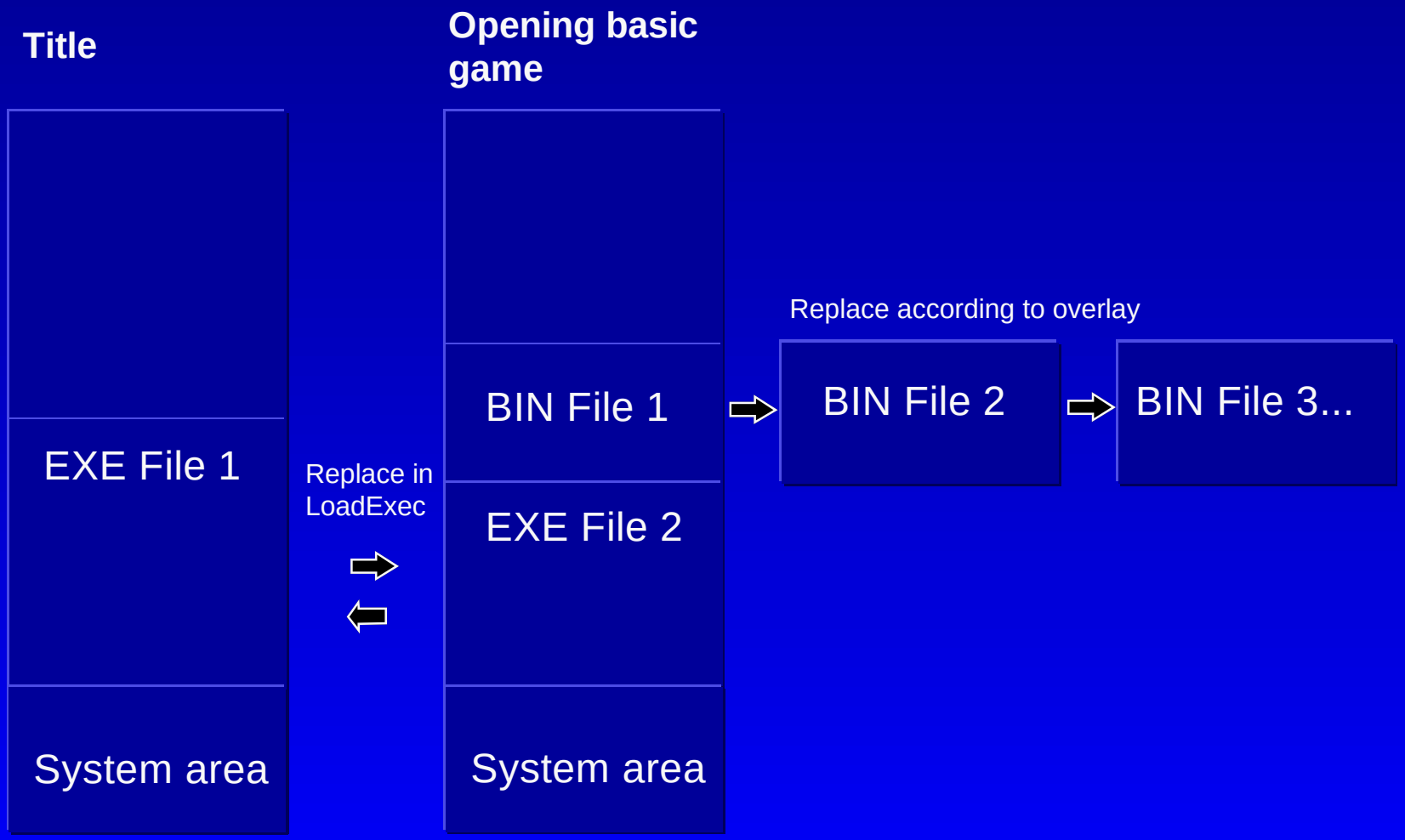


*where OG is the name of the overlay group

Overlay module debugging

- Compile option -g
  - Necessary for source level debugging
- Linker option /v
  - Necessary for overlay debugging

Overlay application

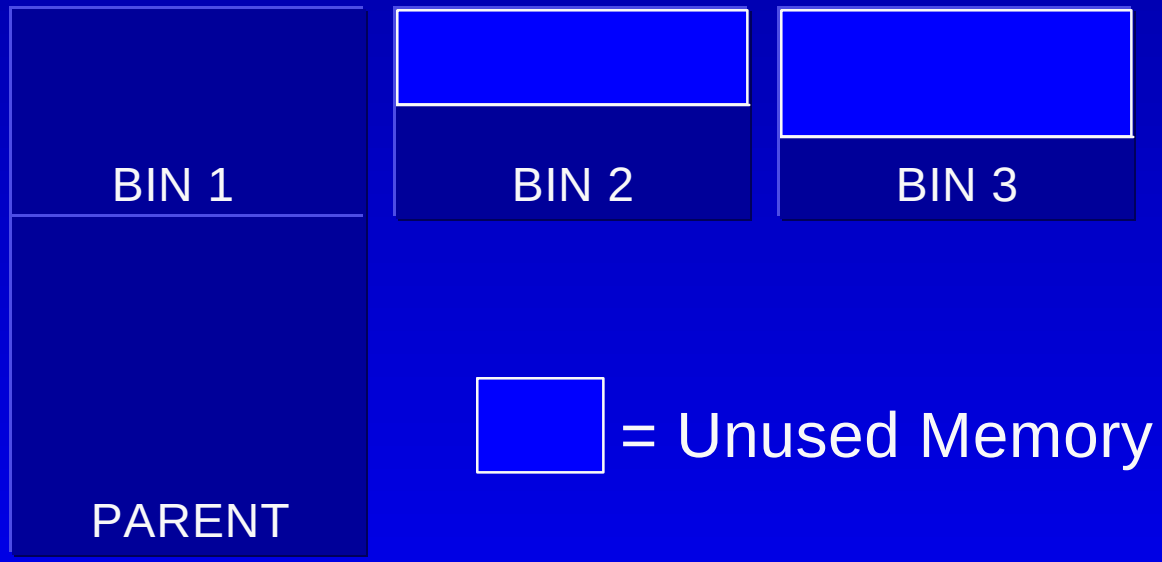


Things to Know About Overlays

Overlay memory usage

Memory reserved for largest overlay

“Lose” memory on smaller overlays



Size and number of overlays

- Smaller overlays:
  - Load faster
  - Require less memory
- Larger overlays can be more complex
- Linker on latest tools CD supports up to 256 overlays

Pitfalls

- Be sure to compile overlays with the -G0 option
- Flush I-cache when loading an overlay
- Keep heap base clear of overlay space
- Overlays can not be interdependent

THE END