MMU なしプロセッサ用 Linux の 共有ライブラリ機構

大谷 浩司、高岡 正、近藤 政雄、臼田 尚志
株式会社アックス

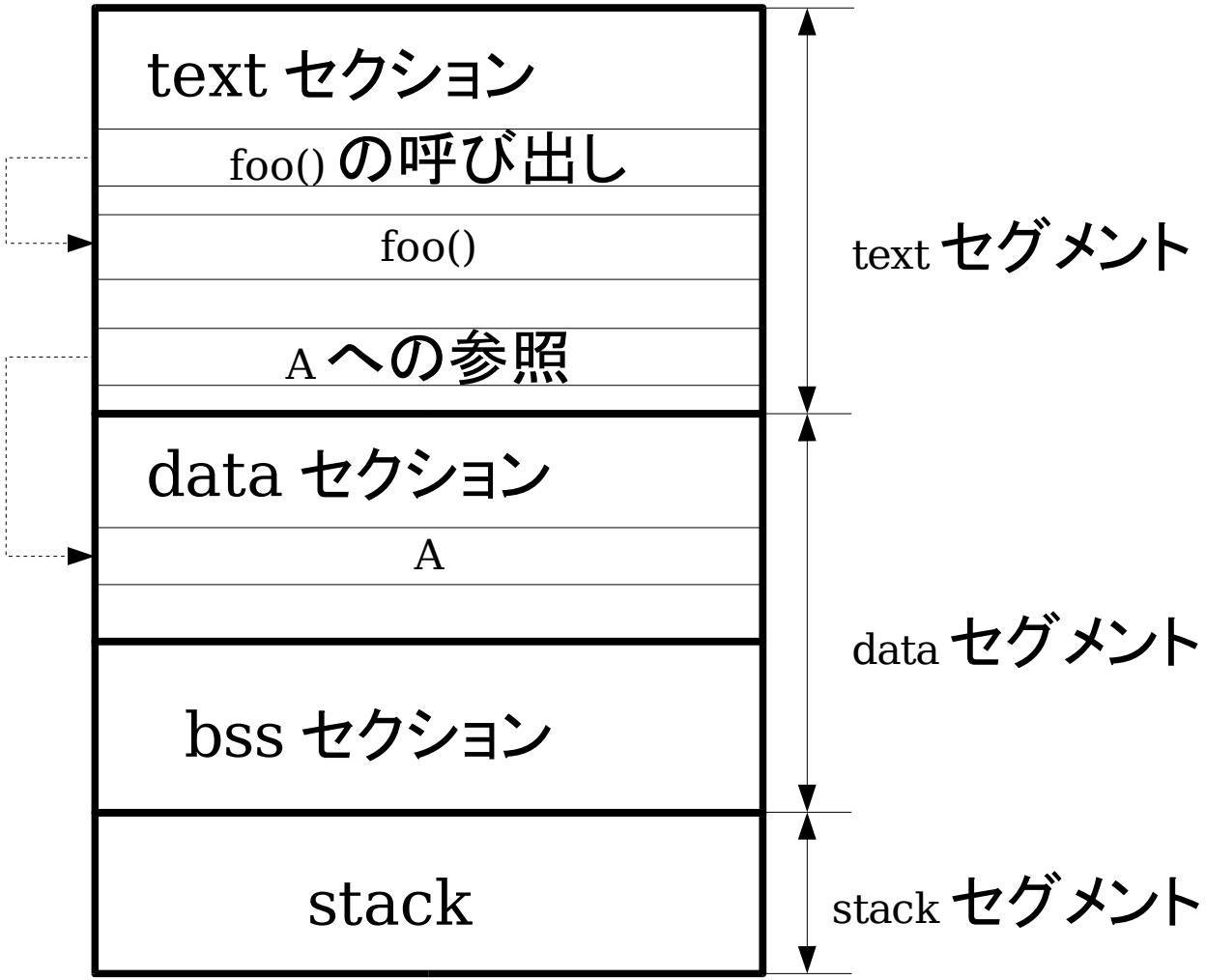
はじめに

- μClinuxには、仮想メモリ機構がないので共有ライブラリ機構が使えない
- でもメモリ消費抑制、ストレージ消費抑制、保守性の向上のためには、欲しい。
- 幾つかの実装があるが、CPU 、ライセンス、機能の制限のためにそのまま利用できない
  - RidgeRun 社（Cadenux 社 ） ARM CPU
  - SnapGear 社　 ColdFire CPU

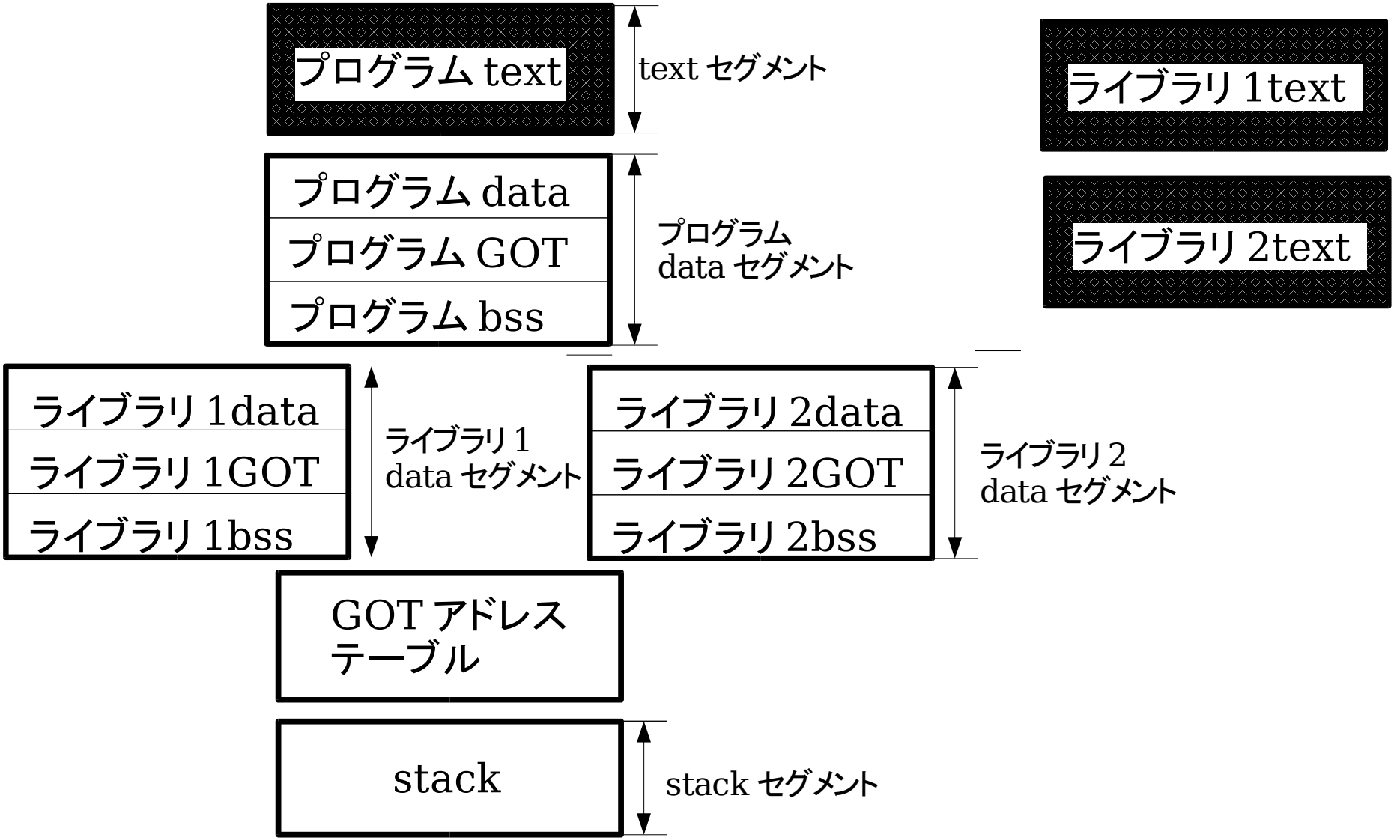
利点

- ライブラリ共有によるメモリ消費抑制
- プログラム・コード共有によるメモリ消費抑制
- ファイル容量削減
- ライブラリを修正しても再リンク不要
- 通常 Linux よりも実行オブジェクトが小さい
- XIP(eXecute In Place) 対応

μClinux でのメモリイメージ



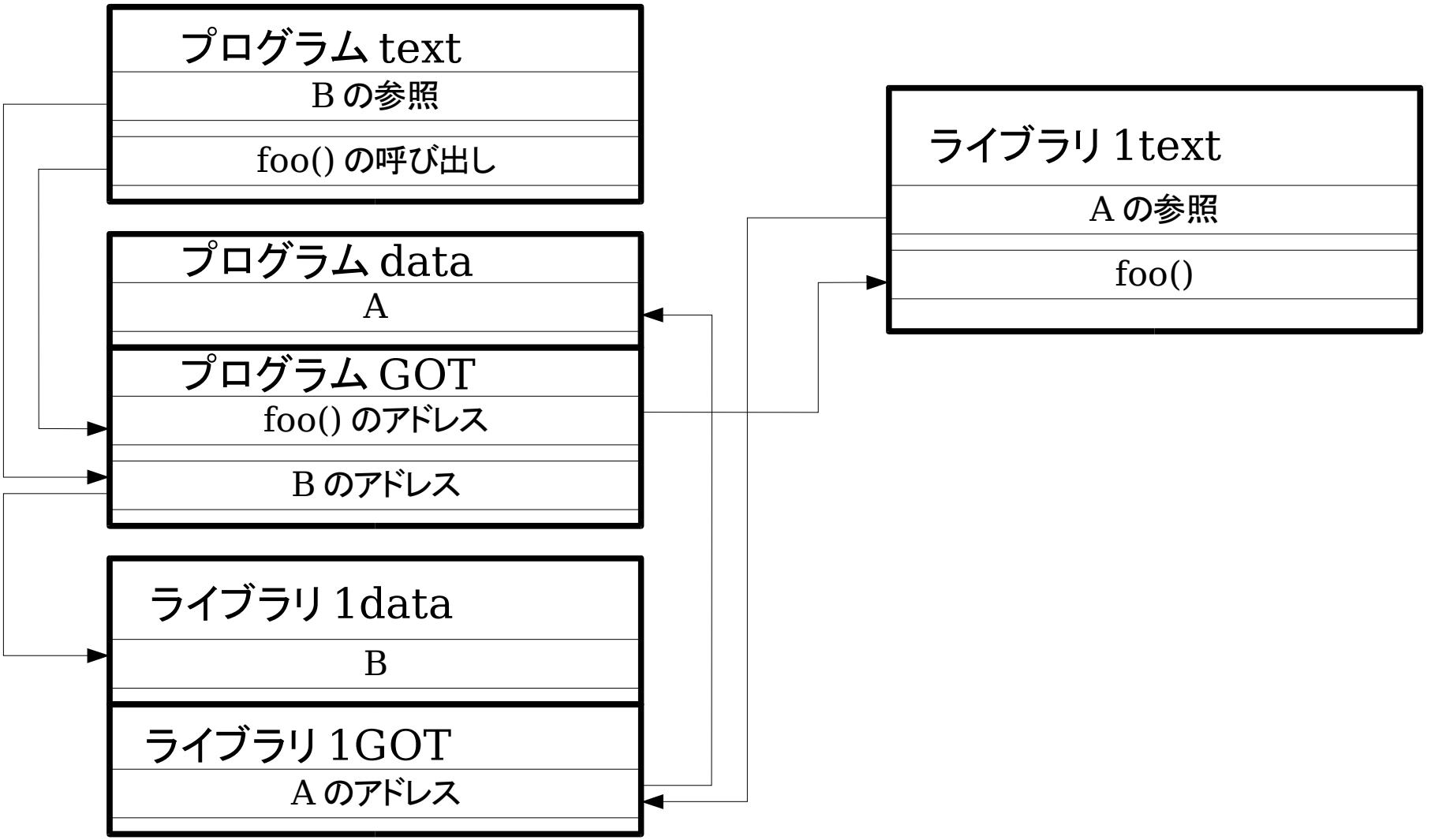
共有ライブラリ機構でのイメージ



GOT(Global Offset Table)

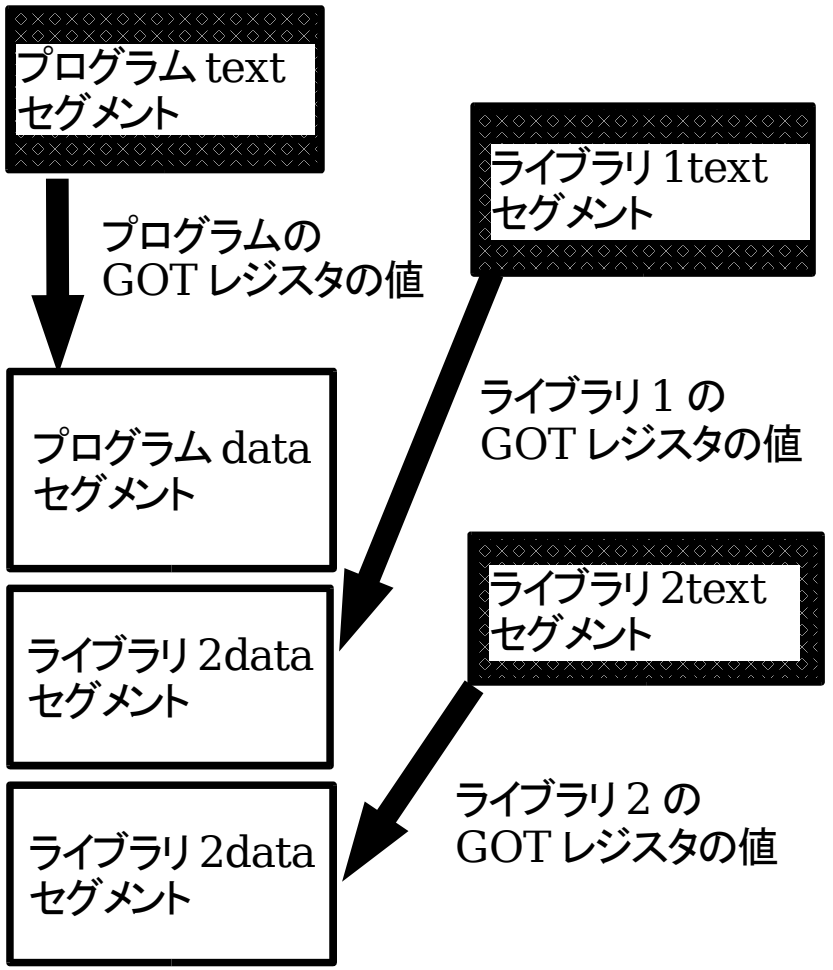
- text セグメントを書き換えないでリンクする手法
- 他のモジュールの関数やデータのアドレスは、ロード時にならないと不明。
- コードから直接参照すると、ロード時にコードを変更する必要がある。
- data セグメントを介して間接的に参照
- このためのテーブルが GOT
- GOT のエントリを正しく設定するのは、動的リンク＆ローダの責任

GOT を使った参照の例



data セグメントのアクセス方式

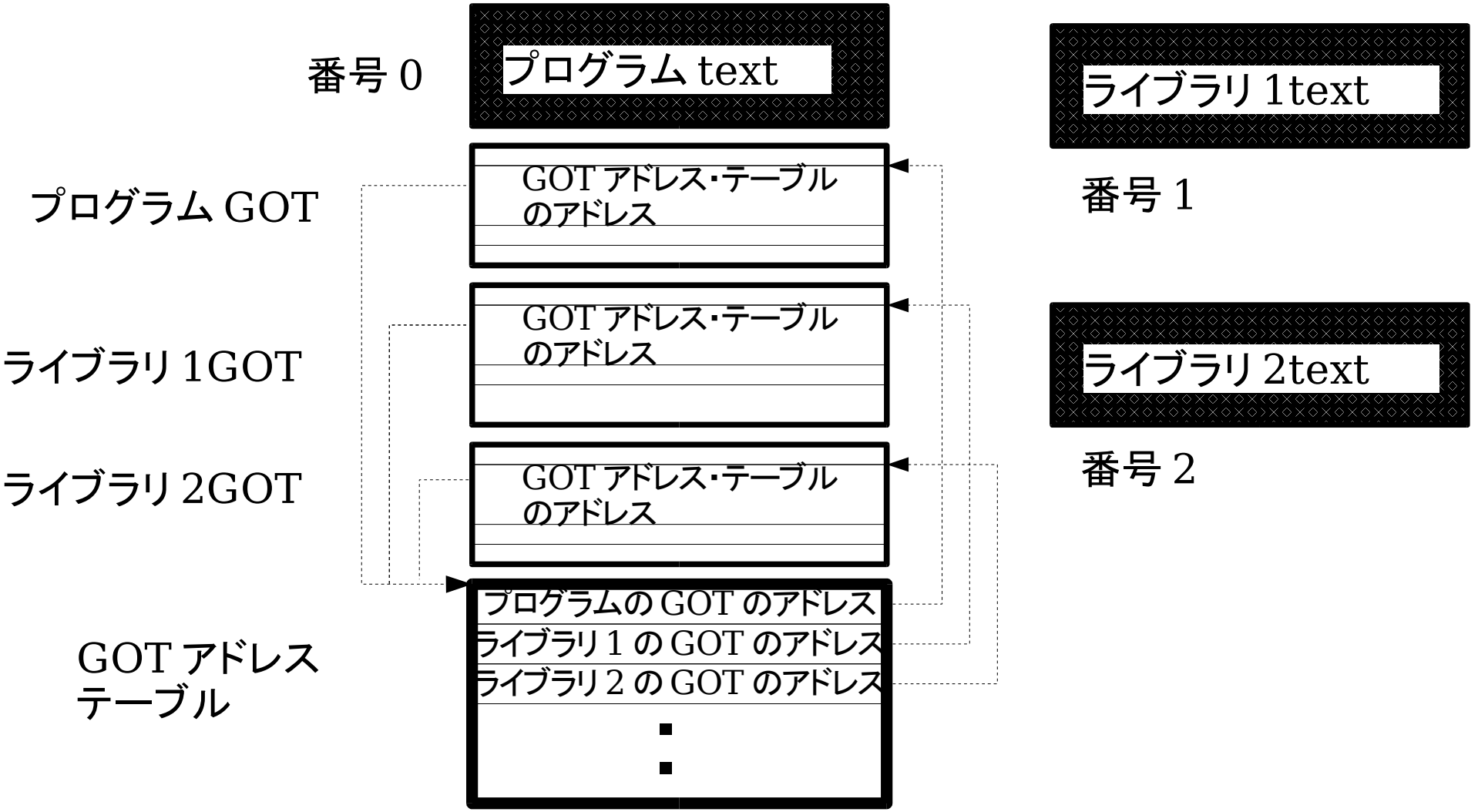
- ベースレジスタ (GOT レジスタ) 相対で参照
- 各モジュール毎に data セグメント
- 各モジュール実行時には、そのモジュールの data セグメントを指すように設定



GOTレジスタの設定方法

- プログラムの起動時のGOTレジスタは、動的リンク＆ローダが設定
- 他のモジュールの関数の呼出時に変更の必要
- 二つの方式
  - 呼び出すモジュールで行なう方式
    - 関数へのポインタを他のモジュールに渡す事ができない
  - 呼び出されるモジュールで行なう方式
- 後者の方式を採用し、関数の先頭部分で設定

自身の GOT 先頭アドレスを知る方法



動的リンカ＆ローダの動作

- text セグメントのロード - メモリ上に存在すれば共有
- data セグメントの確保、初期化
- GOT の初期化
- GOT アドレス・テーブルの確保、初期化
- リロケーション
- インポート・シンボルの解決
- スタックの確保
- 最初の GOT レジスタの設定

text セグメントの共有方法

- PC の Linux では、共有 mmap で実現
- しかし、uClinux では、一般には実装されていない。
- IPC の共有メモリを改造して実現

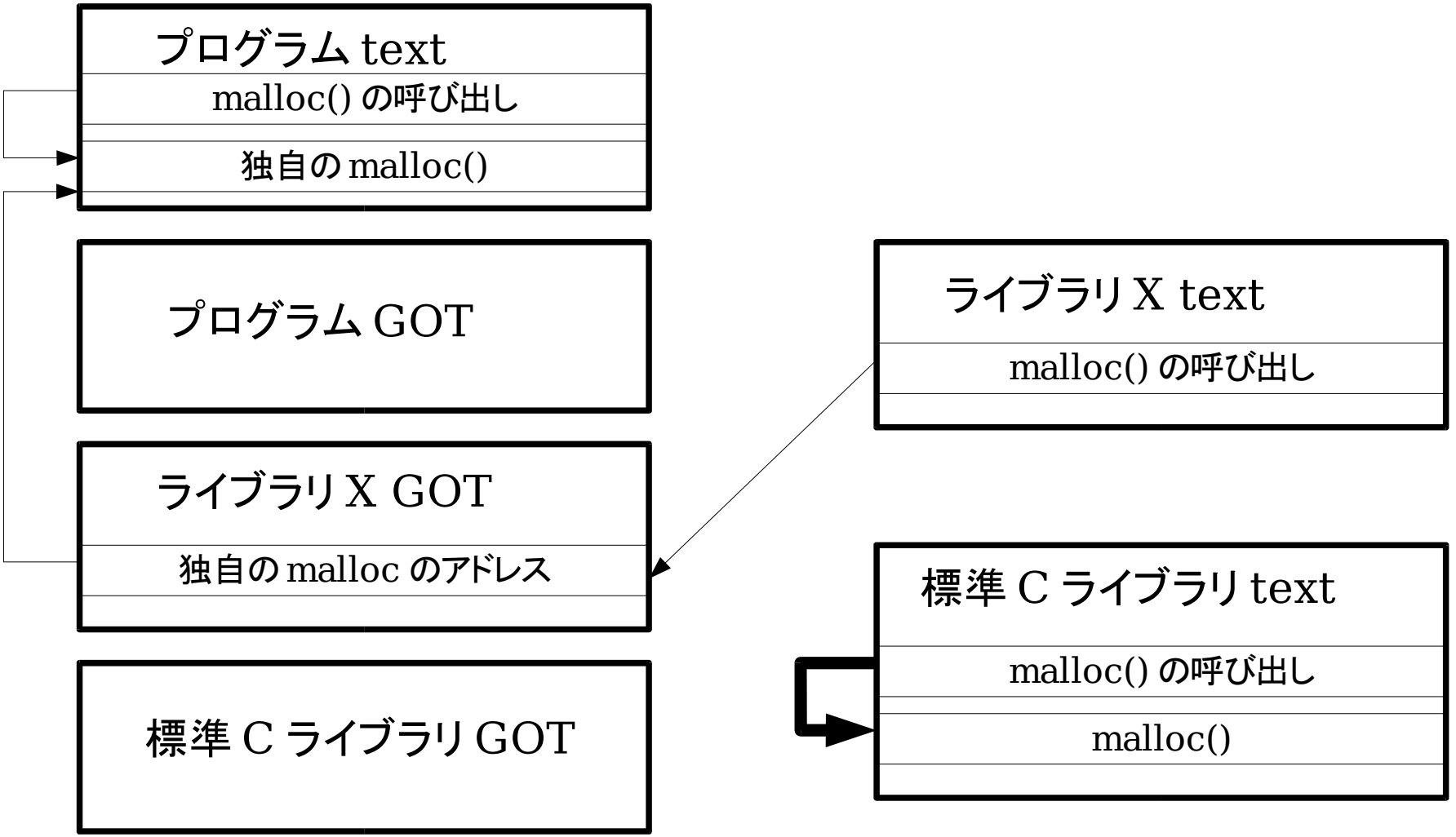
XIP(eXecute In Place)

- プログラムを RAM 上にコピーせずに、ROM 上に置いたまま実行する技術 - RAM が節約可能
- mmap が ROM 上のポインタを返せば、text 共有の機構を使って可能
- romfs で、実験的に実装して、動作を確認

初期化ルーチンの問題

- C++ などでは、main の前に実行すべき初期化ルーチンがある。
  - CTOR セクション
  - __main とCTOR_LIST
- そのままでは、共有ライブラリ側の初期化ルーチンが呼び出されない。
- 共有ライブラリの初期化ルーチンも、いもづる式に呼び出すようにする。

関数の置換えの問題


プログラム text
malloc() の呼び出し
独自の malloc()
プログラム GOT
ライブラリ X GOT
独自の malloc のアドレス
標準 C ライブラリ GOT
ライブラリ X text
malloc() の呼び出し
標準 C ライブラリ text
malloc() の呼び出し
malloc()

既存ライブラリの利用

- 制限付きで既存のライブラリを利用可能
- プログラムの data セグメントを text セグメントに連続してメモリ上に配置することで実現
  - プログラムの text セグメント共有不可
  - 既存のライブラリ中からの共有ライブラリ中のデータアクセス不可
    - GOT レジスタ等の処理が入っていないため

その他の問題

- WEAK シンボル： サポートしない
- COMMON 領域： 制限付きサポート
  - サイズが異なっていては、いけない
- linkonce セクション (C++ テンプレートのインスタンス): 各モジュールにリンクされてしまう。

Cadenux 社の方式

- GOT レジスタに相当する SB レジスタの設定を呼出側のモジュールで行なう。
  - 関数ポインタ渡しの際に問題
    - それ用のマクロや関数を使うようにプログラムを変更する必要がある
    - カーネルの変更が必要
- text 共有に共有 mmap を使用

SnapGear 社の方式 (1)

- リンクをプログラム作成時に静的に実行
  - ライブラリ番号
  - 参照部分には、仮の参照アドレスを設定
    - 上位 8 ビットに被参照モジュール番号：下位 24 ビットにオフセット
    - 動的リンカ＆ローダが実際のアドレスに変換
- 利点
  - 実行オブジェクトが小さく、ロード時の負荷が小さい
- 欠点
  - ライブラリを変更すると再リンクが必要
  - ライブラリの実行時の置換えができない
  - モジュールのアドレス空間が 16M しかない
  - ライブラリをシステム内で最大 255 個しか利用できない

SnapGear 社の方式 (2)

- GOT アドレス・テーブルが、各モジュール毎に存在
  - メモリアクセスが1回減るが、メモリを消費
- text 共有に共有 mmap を使用

本方式でのＸのプログラムの例

- X11perf
  - 共有ライブラリを用いない場合 -1,670,044 バイト
  - 共有ライブラリを用いた場合　-　139,035 バイト
- 標準的ないくつかのプログラムとライブラリのファイルサイズの合計
  - 共有ライブラリを用いない場合 -86.8M バイト
  - 共有ライブラリを用いた場合　-　7.6M バイト
    (ライブラリも含む)

Xserver, ico, oclock, twm, x11perf, xauth, xbiff, xcalc, xclipboard, xcutsel, xclock, xdpyinfo, xedit, xeyes, xhost, xinit, xkill, xload, xlogo, xlsatoms, xlsclients, xlsfonts, xmag, xman, xmessage, xmodmap, xprop, xrefresh, xset, xsetroot, xwd, xwininfo, xwudなどＸのソースツリーに含むもの

まとめ

- uClinux 用の共有ライブラリ機構を開発
- 共有ライブラリ機構には、多くの利点が存在
- GOT 、ライブラリ番号、GOT アドレス・テーブルを利用、ベースレジスタを呼び出された側で設定する方式
- 他の方式と比較しても、利点がある。
- X Window の例では、大きなファイルサイズの抑制効果が得られた。
- 開発に協力して頂いた方々に感謝したい