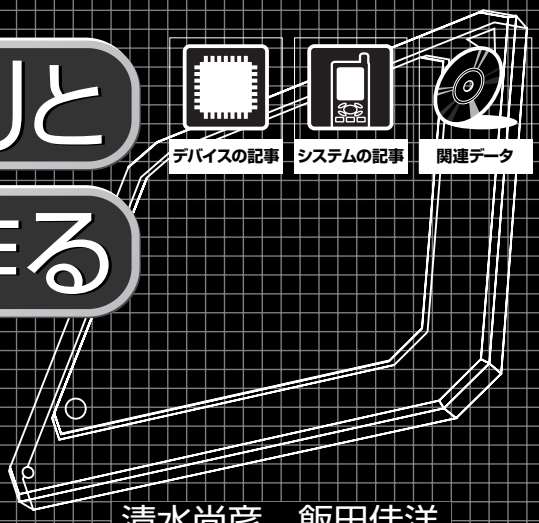
FPGA搭載用CPUとソフト開発環境を作る



第5回
搭載したUNIXのプロセス動作を理解する（最終回）

清水尚彦，飯田佳洋

前回は，拡張命令セット（EIS）とメモリ管理ユニット（MMU）を搭載したCPUの開発について説明しました．今回はそのメモリ管理ユニットの利用例として，UNIXのプロセス動作について解説します．また，本誌2003年10月号に付属していたFPGA（Cyclone）ボードに，本連載で開発したCPUと周辺回路を実装し，実際に動作させてみます．（筆者）

今回は16ビットUNIXを例として，仮想メモリ（アドレス変換）やマルチタスク（マルチプロセス）のプログラム動作について解説します．本連載の第2回（本誌2003年8月号，pp.127-136）で取り上げたリアルタイムOS「proc」は実メモリだけを用いて実装しましたが，UNIXでは基本的に仮想メモリを用いています[注1]．仮想メモリを用いると，ユーザ・プログラムのそれぞれに独立したアドレス空間を与えることができます．アドレス空間を独立させることで，C言語などの高級言語が扱うプログラム・モデルを簡単に実現できます．

例えば，C言語ではコード領域，データ領域，スタック領域という三つの領域を扱います．アドレス空間が独立していれば，コンパイラはほかのプログラムの実行アドレスを気にせずに，みずからのプログラムのアドレス管理だけを考えてコンパイルできます．各ユーザ・プログラム（各プロセス）に独立したアドレス空間を与えるUNIXは，アクセスできる資源を限定した仮想的な計算機をプロセスごとに与えている，とみなすことができます．その結果，プロセス間の独立性は高くなり，セキュリティの向上にもつながります．

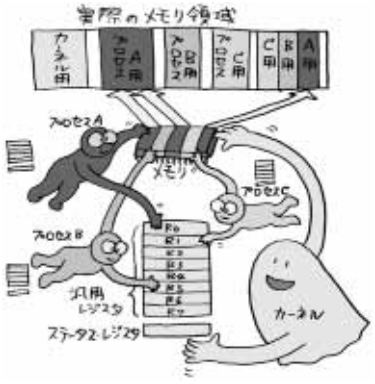
なお，記事内で紹介しているUNIX V6のソース・コードはCaldera（現SCO，本記事中は旧名のまま使用）社のライセンスに基づいて自由に利用できますが，著作権はCaldera社にありますのでご注意ください（詳しくはp.135のコラム「UNIX V6のライセンスについて」を参照）．

注1：UNIX V6（Version 6）には，仮想メモリを用いずにサイズをコンパクトにした「mini-UNIX」という実装もある．

1.各プロセスが利用するメモリとレジスタの確保

C言語が提供するプログラム・モデルを実行するために必要なものは，メモリとレジスタです．これらを，プロセスごとに独立して所有しているかのように見せかける必要があります（**図1**）．

レジスタは数が少ないため，プロセスを切り替えるときにすべてを入れ換えることも可能です．一方，メモリはそう簡単には入れ換えることができません．そこで，実際のメモリ内のデータは入れ換えず，物理アドレスと，ソフトウェアから見える仮想的なメモリ・アドレス（論理アドレ



〔図1〕メモリとレジスタをプロセスごとに用意しているように見せかける

各プロセスからは，自分がメモリ空間（64Kバイト）とレジスタ（R0～R7，およびステータス・レジスタ）を専有しているように見える．

ス）の対応関係を実行時に変換します．本連載で開発したUNIX対応のCPU（PDP11/40相当）には，論理アドレスを物理アドレスに変換するためのテーブル（セグメント・レジスタ）をユーザ・モード用とカーネル・モード用にそれぞれ実装しました．詳しくは，本連載の第4回（2003年11月号，pp.143-150）を参照してください．

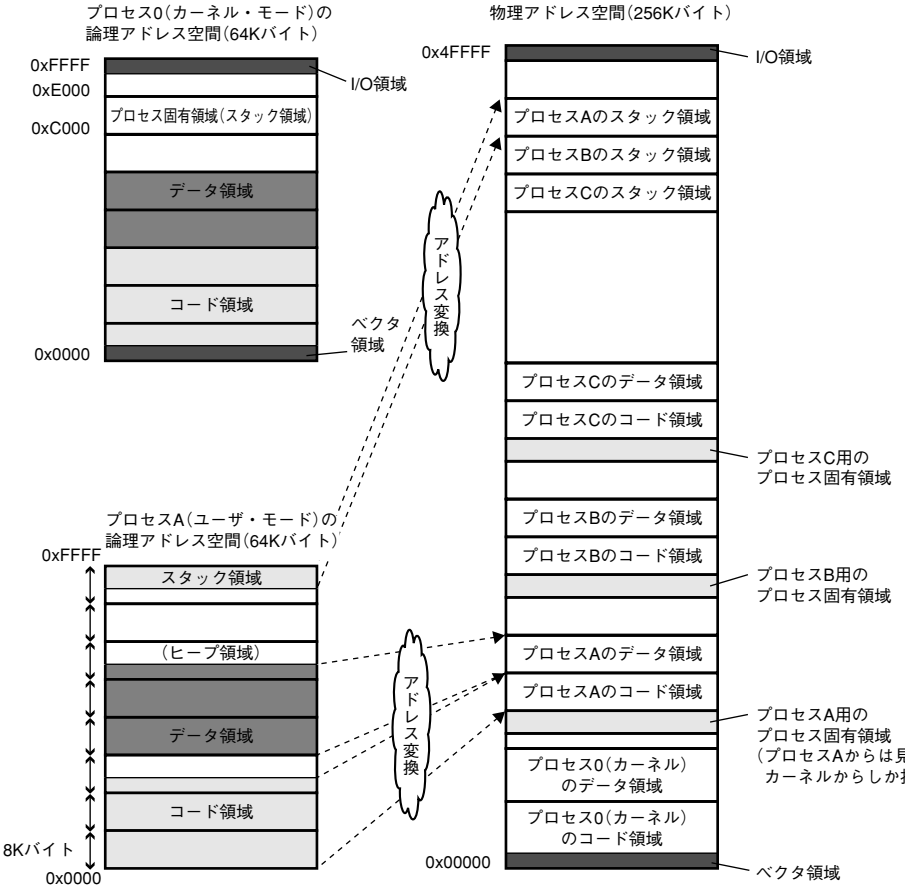
仮想メモリの実現方法としては，ページング（物理メモリをページと呼ばれる固定長のブロックに区画化し，仮想メモリと対応づける方法）とセグメンテーション（物理メモリをセグメントと呼ばれる任意のサイズの領域に区画化し，仮想メモリと対応づける方法）が有名ですが，PDP11/40はその中間的な方法を採っています．

形式としては8Kバイトを1ページとするページングなのですが，セグメント・レジスタ内のページ記述レジスタ（PDR）にてページ長を指定することができます．つまり，可変長のセグメントとして扱えるのです．さらに，メイン・メモリ上に64バイト単位でページを配置できます．つまり，ページ番号であるアドレス空間の上位3ビットをセグメント選択に用いるセグメント方式と考えられます[注2]．

●アドレス変換により256Kバイトのメモリを扱う

PDP11/40では，一つのプロセスが扱える最大のアドレス空間は64Kバイト（16ビット空間）です．これに対して，実装したメモリ（物理アドレス空間）は最大256Kバイトです．この物理アドレスを利用するために，アドレス変換を利用しています（**図2**）．x86プロセッサに実装されている物理アドレス拡張（PAE；Physical Address Extensions）もこれと同じ機能です．

なお，このようにプログラムから直接処理可能なアドレス空間を超えるサイズのメモリを搭載している場合，OSの



〔図2〕
プロセスAを実行している場合（ユーザ・モード）の論理アドレス空間と物理アドレス空間のマッピング
プロセス0（カーネル・モード）の論理アドレス空間も参考として示す．

注2：コンピュータ・アーキテクトの立場からすると，アドレス空間の一部に色を付けるようなこの種の方式は，あまりスマートとは言えないと思う．しかし，PDP11アーキテクチャが設計された時点では大容量のメモリが使えるわけではなかったため，妥当な選択だったのだろう．実はPowerPCやIA64もアドレス空間の上位ビットを用いてセグメントを切り分ける方式を採用している．いまだにこういった設計が好きなアーキテクトは多いようだ．