日曜研究室

技術的な観点から日常を綴ります

[xv6 #8] Chapter 1 – The first process – Paging hardware

テキストの17〜18ページ

本文

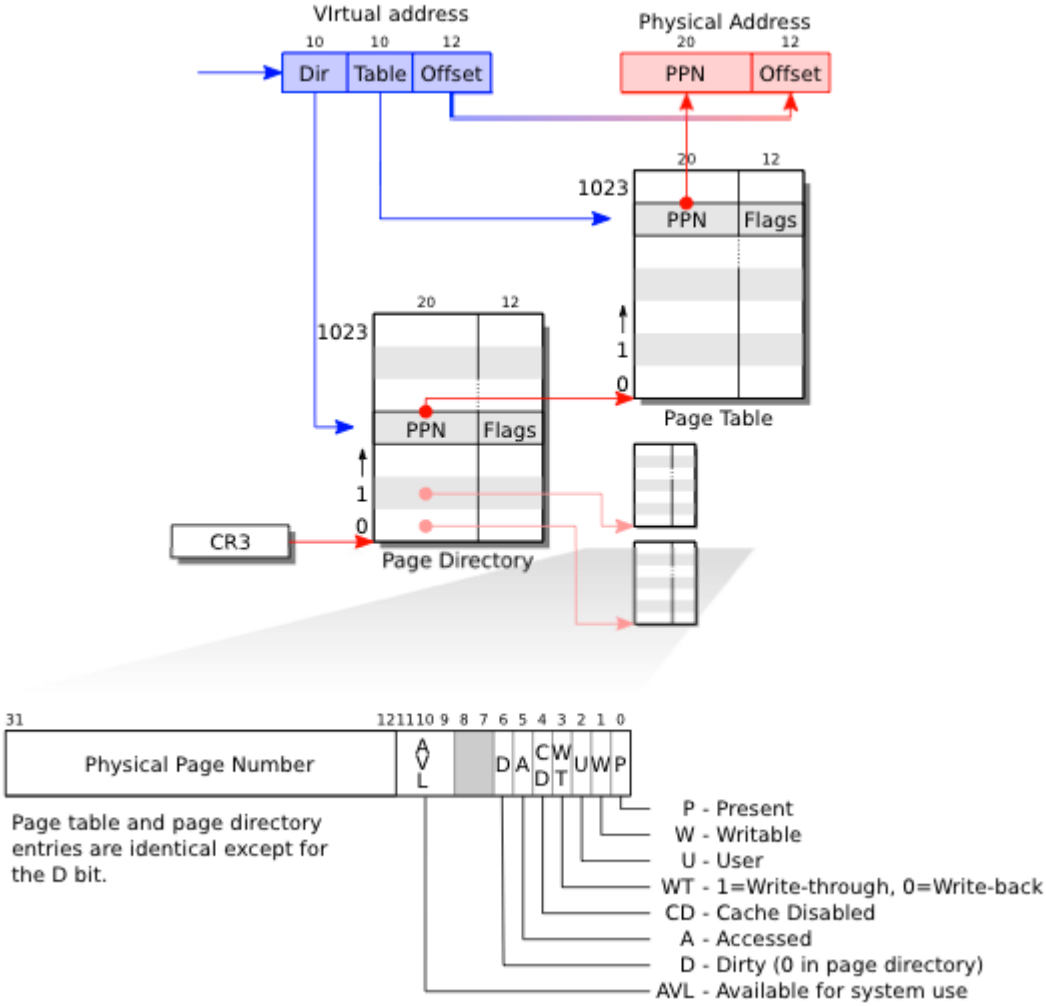
xv6はIntel 80386以降のx86プロセッサを積んだPCで動作する。
そしてxv6にある多くのローレベルの機能（例えば仮想メモリの実装など）は、x86専用である。
このテキストでは、読み手が何らかのアーキテクチャ上でマシンレベルのプログラミングを少しやったことがあると仮定し、x86固有のアイデアを説明する。
（アセンブリ言語を使った事がある人を想定してるみたいです。）
付録Aに、PCプラットフォームの概要について手短に説明してある。
（付録Aは、元のテキストの最後のほうに付属してます。）

x86のページングハードウェアは、仮想アドレス（virtual address）を物理アドレス（physical address）に変換（もしくはマップ）するために、ページテーブルを使う。
仮想アドレスは、x86命令が操作するアドレスで、物理アドレスは、プロセッサチップからメインメモリに送られるアドレスである。

x86のページテーブルは、論理的な2^20（1,048,576）個のページテーブルエントリ（page table entries: PTE）の配列である。
どのPTEも、20ビットの物理ページ番号（physical page number: PPN）といくつかのフラグを含んでいる。
ページングハードウェアは、仮想アドレスの上位20ビットをインデクスとしてそれに対応するPTEをページテーブルの中から検索し、見つかったPTEに保持されているPPNでその仮想アドレスの上位20ビットを置き換える事によって、仮想アドレスから物理アドレスへ変換する。
ページングハードウェアは、下位12ビットをそのまま仮想アドレスから物理アドレスへコピーする。
従ってページテーブルは、4096（2^12）バイトの境界をもつ塊ごとの仮想-物理アドレス変換機能をOSに提供する。
そのような塊はページと呼ばれる。

図1-1にあるように、実際の変換は2段階で行われる。
ページテーブルは、2層のツリーとして物理メモリに格納されている。
ツリーのルートは、4096バイトのページディレクトリ（page directory）である。
ページディレクトリは、ページテーブルのページを参照するためのPTEのようなものを1024個含む。
どのページテーブルのページも、1024個の32ビットのPTEを含む。
ページングハードウェアは、ページディレクトリの中のエントリを選ぶために、仮想アドレスの上位10ビットを使う。
そのページディレクトリのエントリが有効なら、ページングハードウェアは、そのページディレクトリのエントリが指し示すページテーブルのページを参照し、仮想アドレスのの次の10ビットを使い、そのページテーブルからPTEを選択する。
もしページディレクトリのエントリやPTEが無効なら、ページングハードウェアは違反（fault）を発生させる。
（多分これがページフォルトなのかな）
この2層の構造は、仮想アドレスの広い範囲がまだマップされてないようなよくあるケースで、ページテーブルに対してページ全体を省略可能にする。

図1-1 x86上のページテーブル



どのPTEも、関連付けられた仮想アドレスが利用可能かをページングハードウェアに伝えるための、フラグビットを持っている。
PTE_Pフラグは PTEが有効かどうかを示す。
もしそれがセットされていなければ、そのページに対する参照は違反（fault）を起こす。
PTE_Wフラグは、そのページに対する書き込み命令の発行が許されているかどうかを制御する。
もしそれがセットされていなければ、データもしくは命令の読み込みだけが許可される。
PTE_Uフラグは、そのページがユーザープログラムで使用可能かどうかを制御する。
もしそれがクリアされていれば、そのページはカーネルのみ利用可能となる。
図1-1にPTEの全てのフラグが示されている。

用語の補足。
物理メモリとは、DRAM上のストレージセルのことを指す。
物理メモリは1バイトごとに、アドレス（物理アドレスと呼ばれる）を持つ。
プログラムは仮想アドレスを使用する。
仮想アドレスはページングハードウェアによって物理アドレスに変換され、（メモリ上の）ストレージを読み書きするためにDRAMハードウェアへ送られる。
（CPU視点では、DRAMに物理アドレスを投げるとその中のデータが返ってくる）
この説明の段階では、仮想アドレスについてのみ述べていて、仮想メモリなどはまだ関係ない。

感想

この節を一言で表すと、「4KBごとに分けられたメモリ領域をハードウェアがどう管理するかという事の説明」かと思います。

今回はまだ、メモリ管理ユニット – Wikipediaを見ながら、まぁ理解できたかなと思います。

ページが4KBでそれが2^20個まで使えます。
それを計算すると4GBとなります。
32ビットOS・CPU（物理アドレス拡張なし）で使えるメモリの最大量は4GBでしたね。

アセンブリ言語の使用経験を要求されてるようですが、10年以上前にほんの少しやっただけで、今では綺麗サッパリと忘れてしまってます。
これから先大丈夫だろうか(; ´Д `)

カテゴリー: 技術 l タグ: xv6 l 投稿日: 2012/2/14 火曜日 [http://peta.okechan.net/blog/archives/1253] l