第 6 回の課題について

（1）仮想アドレス 20716 番地に対する仮想ページ番号と、ページ内オフセット値は、ページサイズが ① 4KB (4096 バイト) のとき、 ② 8KB (8192 バイト) のとき、それぞれどうなるか

アドレス値をページサイズで除算すると、商が最終ページ番号 (0 ページ開始) で余りが最終ページ内でのオフセットとなりますので、以下のようになります。

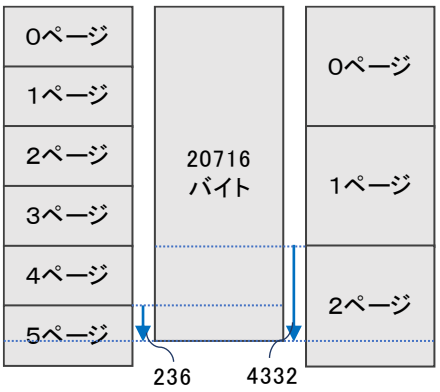
① ページサイズが 4096 バイトの場合

20716÷4096＝5…236 → 5 ページのオフセット 236

② ページサイズが 8192 バイトの場合

20716÷8192＝2…4332 → 2 ページのオフセット 4332

(①の結果より、4KB の 0,1 ページが 8KB の 0 ページ、4KB の 2,3 ページが 8KB の 1 ページで、2 ページのオフセットは 4096+236 ともできます)




```
Python 3.10.12 (main, Nov 20 2023,
Type "help", "copyright", "credits"
>>> 20716//4096, 20716%4096
(5, 236)
>>> divmod(20716, 8192)
(2, 4332)
```

計算間違いの方も見受けられました。

（2）ページングを実装する場合のオーバーヘッドにはどのようなものがあると考えられるか

ページングでの問題点を時間の点と空間の点から考えますと、

- 時間的なもの … メモリアクセスの増加

ページングではページテーブルがメモリ内に存在するので、プログラムでメモリをアクセスする際に必ずメモリ内のページテーブルをアクセスして、ページテーブルエントリの内容 (実ページ番号) を得、それによって再度メモリをアクセスすることになります。つまり、一度のメモリアクセスに対して 2 回のメモリアクセスが必要になります。(多段のページテーブルをもつことにすると、さらにその分メモリアクセスが増えます)

メモリアクセスは、CPU の命令実行に対して 100 倍程度時間を要するので、メモリアクセスの増加は時間オーバーヘッドにつながります。

この問題に対する解決策は、直近に行われた「仮想ページ → 実ページ」の変換結果を CPU 内の高速メモリ (今回の補足の TLB) に保存して再利用しようとするものです。アクセスするページ番号が TLB 内にあったときは、実ページ番号が即座に得られます。

プログラムがアクセスするメモリアドレスは連続することが多いので、ページ番号部が同じであることが期待できます。したがって最初に TLB エントリが設定されると、以降のアクセスは

そのエントリを再利用でき、メモリ内のページテーブルへのアクセスが大幅に削減できることになります。
　また、ページフォールトに対する例外処理も時間的なオーバーヘッドになります。

　さらに、デマンドページングと組み合わせて仮想記憶を構成すると ページイン・ページアウトでの二次記憶アクセスに要する時間が大きなオーバーヘッドになります。

- 空間的なもの … ページテーブルサイズの増加

　ページテーブルはメモリ内に置くもので、講義でも説明しましたが、4 KB ページで 32 ビットの仮想アドレス空間の場合、100 プロセスで 400MB などと無視できないサイズになります。
　この問題に対する解決策として、講義資料の多段のページテーブルがあります。また、ハッシュ関数を用いてページテーブルを管理するなどの方式もあるようです。

【提出いただいた解答へのコメント】
（1）では「5 ページのオフセット 236」といった数値だけの解答の方もいらっしゃいますが、「アドレス値をページサイズで除算すると、商がページ番号で余りがページ内オフセットとなる」や「20716÷4096＝5…236 → 5 ページのオフセット 236」など、解の求め方についての記述があるものが望ましいと考えています。

　20716 を二進表現すると 00… 0101 0000 1110 1100 となります。

　ページサイズサイズが 4096 (2^{12}) バイトの場合、最下位ビット (ビット 0) から 12 ビット分がページ内オフセット ($(0000\ 1110\ 1100)_2 = 236$) で、12 ビット目以降がページ番号 ($(0101)_2 = 5$) になる。

31　　12 11　　0
00… 0101 0000 1110 1100

　8192 (2^{13}) バイトの場合は、最下位ビットから 13 ビット分がページ内オフセット ($(1\ 0000\ 1110\ 1100)_2 = 4332$）で、 13 ビット目以降がページ番号（$(010)_2 = 2$） となる。

31　　13 12　　0
00… 0101 0000 1110 1100

として解答いただいた方もいらっしゃいました。

（2）ではフラグメンテーション (断片化) について言及していただいた方もいらっしゃいました。ページングでは外部断片化は発生しませんが、内部断片化はあります。ただし、最終のページに発生するだけですので十分小さいと考えることができるというものです。