SMSW.
Загрузка и восстановление состояния блоков **FPU, MMX, SSE** операция требующая много времени. В целях производительности эти блоки сохраняются только если они используются потоком. Далее рассматривается только **FPU**. Механизм оптимизации следующий.
Регистр управления **Cr0** содержит три бита, определяющие поведение процессора при обращении к **FPU**. Планировщиком используются биты **TS(Task Switched)** и **MP(Monitor Coprocessor)**[***Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3A System Programming Guide, 2.5***].
Из всех возможных комбинаций используются две - эти биты установлены, либо сброшены. Если биты сброшены, инструкции **FPU** не вызывают исключения.
Если взведены, то исполнение инструкций блока **FPU, WAIT/FWAIT** приводит к генерации исключения не присутствующего устройства(**#NM**). Обработчик этого исключения(**KiTrap07**) сбрасывает в регистре **Cr0** эти два бита, что является признаком обращения к **FPU**. После чего выполняется рестарт инструкции и последующие обращения к сопроцессору не генерируют исключения. Планировщик(**SwapContext**) при переключении на новый поток проверяет эти два бита в регистре **Cr0** и если они сброшены сохраняет состояние сопроцессора(**fxsave, NPX** - фрейм в ядерном стеке) и взводит биты **TS** и **MP** в регистре **Cr0**. Тоесть после первого обращения к **FPU** возникает исключение, обработчик сбрасывает биты в **Cr0** и далее код выполняется без исключений до окончания кванта времени, отведённого потоку. На следующем кванте обращение к **FPU** снова генерирует исключение.
Младшая часть регистра **Cr0**(**Mashine Status Word**) может быть считана инструкцией **smsw**(доступна на всех уровнях привилегий, [***Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 2B Instruction Set Reference***]), интересующие нас поля(остальные постоянны, маска **110001B**):

```
    TS EM MP PE
... X  0  X  1
```

Биты **PE** и **TS** одинаковы, тоесть регистр может принимать значения **0x31** и **0x3B**(в ядре маска определена как **NPX_STATE_NOT_LOADED = 1010B**). С помощью этой инструкции могут быть считаны эти два бита. Это позволяет определить начало кванта:

```
CR0_MP equ 0010B
CR0_TS equ 1000B

; +
; Ожидание начала кванта.
; В течении следующего кванта высокая вероятность что:
; o Процессор не будет переключен на другой поток.
; o Поток не будет переключен на другой процессор.
; Данный код должен использоваться перед выполнением
; кода в потоке, который не должен быть переключен на
; другой процессор или процессор отдан другому потоку.
;
WAIT_NEXT_QUANTUM macro
       ; Если в текущем кванте небыло обращений к FPU,
       ; генерируется #NM, флажки TS и MP сбрасываются.
       ; Иначе инструкция пропускается.
       fwait
@@:
       ; Читаем MSW. Бит TS будет установлен на следующем кванте.
       smsw ax
       ; Цикл ожидания установки бита TS.
       test ax,CR0_TS
       jz @b
endm

; +
; Определяет использовался ли сопроцессор на последнем кванте.
; ZF = 1 если использовались.
;
IS_NPX_CALLED_ON_LAST_QUANTUM macro
       smsw ax
       test ax,CR0_TS
endm
```

Юзермодный поток может быть вытиснен в любой момент, поэтому приведённый код снижает вероятность переключения, но не исключает такую возможность. Посредством ожидания начала кванта можно определить число переключений контекста:

```
UsTickCountLow             equ 7FFE0000H
UsTickCountMultiplier      equ 7FFE0004H

GET_TICK_COUNT macro
       mov eax,ds:[UsTickCountLow]
       mul dword ptr ds:[UsTickCountMultiplier]
       shrd eax,edx,18H    ; Milliseconds.
endm

WAIT_NEXT_TICK macro
       mov eax,ds:[UsTickCountLow]
@@:
       cmp ds:[UsTickCountLow],eax
       je @b
endm
```

```
WAIT_TIME    equ 1000     ; Ms.

       xor ebx,ebx
       ; Ожидание обновления счётчика тиков.
       ; Начало измерения на новом тике.
       WAIT_NEXT_TICK
       GET_TICK_COUNT
       lea esi,[eax + WAIT_TIME]
Swap:
       ; Сброс TS.
       fwait
@@:
       ; Проверка на таймаут.
       GET_TICK_COUNT
       cmp eax,esi
       jnb @f
       ; Цикл ожидания следующего кванта.
       smsw ax
       test ax,CR0_TS
       jz @b
       inc ebx
       jmp Swap
@@:
       mov eax,ebx
       int 3
```

При дефолтном приоритете поток свопится **~110** раз в секунду. Эта величина колеблется в зависимости от загрузки системы и уменьшается при повышении приоритета. Оценить вероятность переключения контекста можно следующим образом:

```
       xor ebx,ebx
Swap:
       ; Ожидание нового кванта.
       WAIT_NEXT_QUANTUM
       ; Тут низкая вероятность переключения контекста.
       ; Прерывания и исключения не инициируют переключение
       ; контекста до окончания кванта.
       ; Сброс TS.
       fwait
       inc ebx
       ; Проверяем TS. Взводится если контекст был переключен.
       IS_NPX_CALLED_ON_LAST_QUANTUM
       jz Swap
       mov eax,ebx
       int 3
```

При нормальном приоритете данный код выполняется десятки секунд(тысячи циклов), пока не произойдёт переключение контекста на новом кванте. Код без ожидания нового кванта практически сразу выходит из цикла(доли секунды). При высоком приоритете вероятность переключения контекста на новом кванте крайне мала и ей можно пренебречь.

Июль 2009, virustech.org