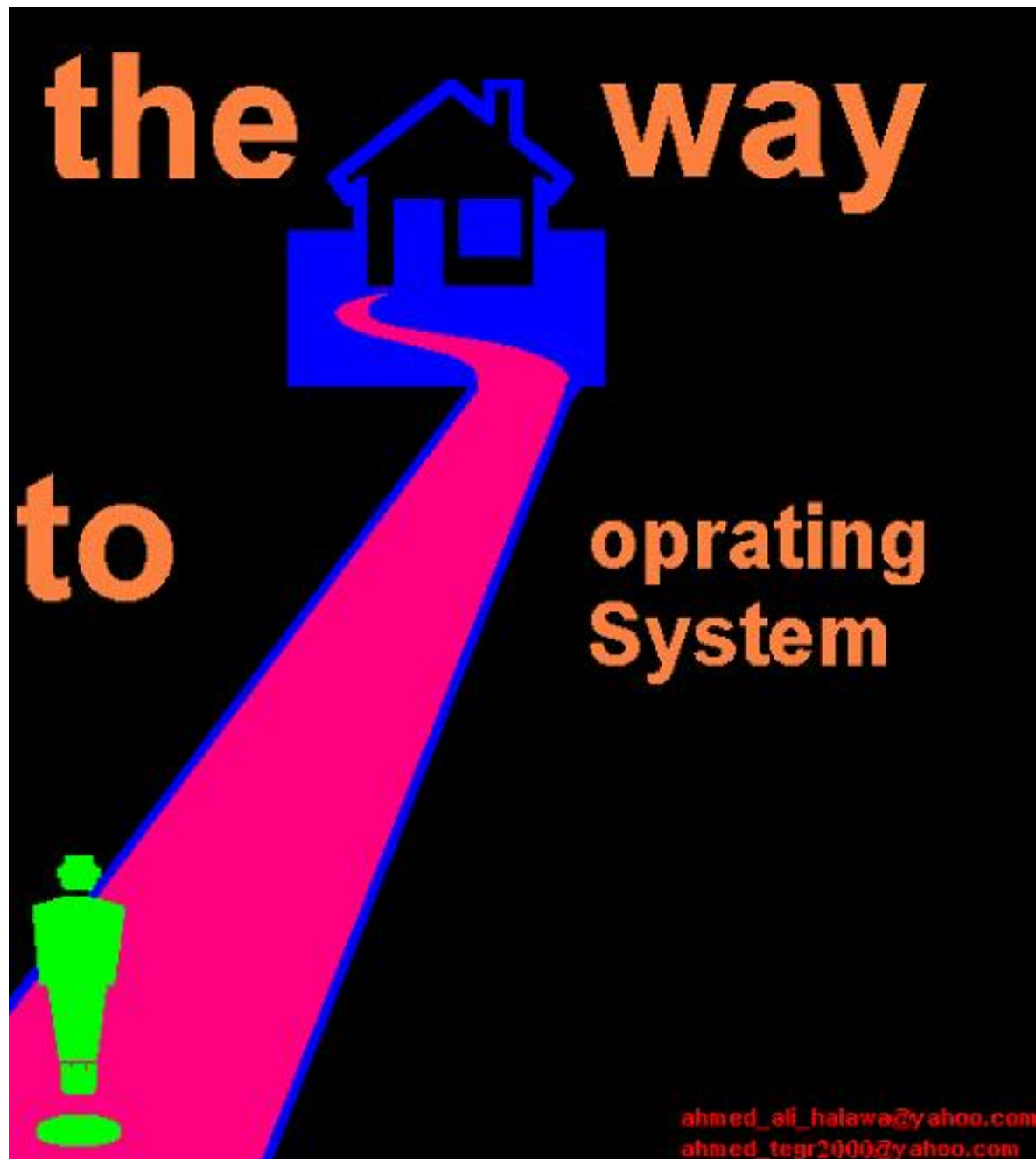




الدرس الرابع: " الذاكرة الوهمية"

الصفحة

العنوان

▷ تعريف بالذاكرة الوهمية ◁

اولاً وقبل تعريف الذاكرة الوهميه بأنتي لتعريف الذاكرة عموماً الذاكرة هي عنصر سلبى لا يؤثر بشئ مهمته تخزين المعلومات وإرجاعها عند الاحتياج وهناك نوعان للذاكرة

← | **الذاكرة الحقيقية :** وهي المثبتة علي اللوحة الام وهي احجام مضاعفه

تبدأ ب ٣٢ ثم ٦٤ ، ١٢٨ ، ٢٥٦ ثم ٥١٢ ثم ١٠٢٤ ميجا

الذاكرة الوهمية هي ذاكره اضعفه للذاكره الوهميه احجامها ليست

ثابتہ ولیست مضاعفہ قد تګون مخزنہ علی قرص مرن او قرص

صليب المهم انه ليس لها علاقه باللوحه الام وقد تصل احجام هذه

الذاكره إلي جيجا بايت.

لما الذاكره الوهميه:

نلاحظ في انظمة ويندوز و linux و نقوم بتشغيل اكثر من برنامج في وقت واحد ولو

ولو اعتبرنا اننا نقوم بتشغيل برنامج حجمه من ٥ إلى ٢٠ ميجا بايت بالإضافة إلي الادوات

التي سيحتاجها وبيئته و..... إلخ سوف يصل حجمه إلي حوالي ٥٠ ميجا في الذاكرة إذا

بالإضافة إلى بيانات النظام و النظام نفسه ماذا سيحدث إذا اردنا تحميل برنامج اخر لم يعد

هناك ذاكره كافيه لذا فإننا بحاجة إلي الذاكره الوهميه

لنفرض انك قمت بعمل dump لبرنامج ما وكان العنوان الذي يعمل فيه هو 0xD00000 وقمت بعمل dump

لبرنامج اخر يعمل ووجد أن البرنامج يعمل في نفس العنوان سوف يخطر ببالك هل كل البرامج تعمل في عنوان

واحد؟ الحقيقة لا فالبرنامج الذي يعمل (عن طريق خاصية تعدد المهام) يتم نقله إلى الذاكرة الحقيقة والآخر

يقيم في الذاكه الوهميه إلي ان يحين دوره هذه افكار بسيطه.

نفرض ان احد البرامج اراد الولوج إلى منطقه معينه من الذاكره وهذه المنطقه ليست موجوده في الذاكره الحقيقيه

فما الذي يسحدث؟

اولاً يتم عمل إنقطاع برمجي تذكر درس الإنقطاعات البرمجي. ويختار الاستثناء رقم ١٤ أي

page fult يعني خطأ في محاوله الولوج فيقوم النظام لابتحث عن هذه المعلومات إن كانت موجودة ام لا ثم يقوم

بتحميلها إلي الذاكرة الحقيقة هذه أشهر الأشياء التي يجب معرفتها

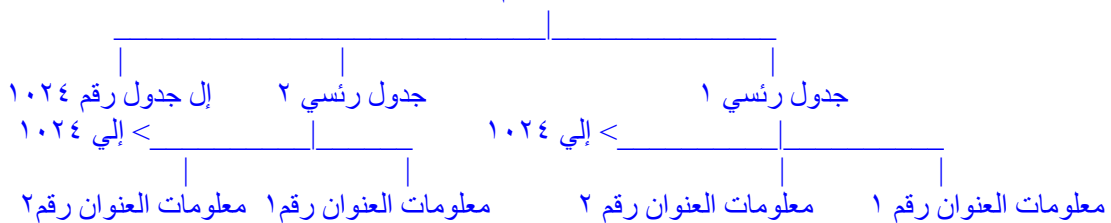
طريقه ترجمه العناوين الإقتراضيه:

اولاً لدينا عنوان البرنامج السابق 0xD00000 في أي منطقه في الذاكره الحقيقه يعمل ؟

يعمل في هذه المنطه 0x10A01B ☺ ماهذا لقد اختلف العنوان كثيراً كيف تحول إلى ذلك

صراحه هناك جداول تسمى جداول الترجمة هذه الجداول كا الاتي

جدول عام



كيف نعرف رقم الجدول الذي يخزن فيه معلومات الجدول وكيف اعرف المنطقة المخزن فيها معلومات الجدول؟

اولاً الطريقه النظرية:

١- دخول العنوان الافتراضي إلى البايت =٠١٠١.....

٢- نأتي بأول ١٠ بايتات من جهة اليسار وهم = ١١٠٠٠٠٠٠٠٠ = 0X3 = الجدول رقم ٣

٣- نأتي بثنائي ١٠ بايتات من جهة اليسار وهم $= 0 \times 100 = ٠١٠٠٠٠٠٠٠٠$ = المنطقة رقم ١٠٠

في الجدول رقم ٣

٤- نأتي بأخر ١٢ بايت في العنوان = 0X0 = = اول منطقه العناوين

سؤال لماذا تم تقسيم منطقہ الترجمة بهذا الشكل ؟

أولاً **معلومه هامه** الجدول العام ليس جدولاً وإنما منطقته في الذاكرة من المعلومات
أخي لننظر إلي الأتي نحن نمك ١٠٢٤ جدول رئيسي يتحتوي كل مدخل (أي منطقة معلومات) علي ١٠٢٤
وكل مدخل يحتوي علي عنوان لصفة (تذكر الدرس السابق) أي نحن نمك لنقم بهذه العملية الحسابية

$$\text{عدد الجداول} \times \text{عدد المداخل في كل مدخل} \times \text{حجم الصفحة بالبايت} = \text{الحجم الذي يستطيع الجدول تحمله}$$

نقوم بالقسمة علي ١٠٢٤ لتحويل الناتج إلي حجم بالكيلو بايت = ٤١٩٤٣٠٤

بالميجا بايت = ٤٠٩٦ " " " " " " " "

" " " " " " " " بالجيڃا بايت = ۴ جيڃا

تذكر درس النمط المحمي اذكر حين قمنا بخل الحجل الذي يتحملة المدخل ٤ جيجا هكذا نكون قد اكملنا ال ٤ جيجا الخاصة بالمدخل

ملحوظة : ال ٤ جيجا تحتوي ايضاً على الذاكرة الحقيقية

سؤال هل للشكل السابق تكوين مداخل مثل جدول المواصفات العلم (جدول المداخل) او جدول الإنقطاعات إن كان نعم فمل تكوين مدخل في الجدول العام وما تكوين مدخل في جدول رئيسي؟
الاجابه

نعم هناك تكوين وهما لا يختلفان كثيراً

مدخل في الجدول العام

٢١	٢٠	١٩	١٨	١٧	١٦	١٥	١٤	١٣	١٢	١١	١٠	٩	٨	٧	٦	٥	٤	٣	٢	١	٠
العنوان الحقيقي وليس الافتراضي للجدول											غير مسجلة	-	P S	.	A	P C D	P W T	U / S	R / W	P	

مدخل فی جدول رُئی

٢١	٢٠	١٩	١٨	١٧	١٦	١٥	١٤	١٣	١٢	١١	١٠	٩	٨	٧	٦	٥	٤	٣	٢	١	٠
المعنوان الحقيقي للصفحة (نذكر الذاكرة الحقيقية)											غير مستعملة	G	P A T	D	A	P C D	P W T	U / S	R / W	P	

$P = 1$ صورہ الصفحة موجودہ فی الذاکرہ الحقیقہ ، صورہ الصفحة موجودہ فی الذاکرہ الوهمیہ

$Rw = 1$ هذه الصفحة للقراءة والكتابة . للقراءة فقط

Us = درجة الافضليه الخاصة بالصفحة

 $\bullet = P_{wt}$ $\bullet = \text{Pcd}$

A = ١ تعنى تم العمل فى هذه الصفحة من قبل احد البرامج

[illegible]

Ps = حجم الصفحات المخزنه فى الجدول ١=٤ ميغا ، ٠ = ٤ كيلو

• =Pat

G = تعنى عدم اخراج معلومات الصفحه من ال tlb

ال TLB

كما رأينا فإن مجرد الولوج إلى موقع في الذاكرة الافتراضية أصبح يتطلب عدة عمليات ولوج وسيطة لمجلد الصفحات و جداول الصفحات. عملية الولوج إلى الذاكرة تعتبر مكلفة بالمقارنة مع سرعة عمل المعالج. من أجل ذلك توجد على الحاسب ذاكرة صغيرة تعمل مثل مخزون cache لترجمة الصفحات. تحتوي هذه الذاكرة على العناوين الحقيقية للصفحات التي تم الولوج إليها مؤخرا. هكذا بدل الولوج إلى جداول الترجمة عند كل ولوج إلى الذاكرة فإن المعالج يقوم أولا بالنظر ما إذا كانت الترجمة متوفرة في مخزون الترجمة TLB. باستعمال هذه التقنية فإن الأداء قد يتحسن بصورة ملحوظة إذا كان البرنامج يستعمل عناوين متقاربة - بلغة تقنية نقول أن البرنامج لديه محلية جيدة Good locality :-

ملحوظة: لا يجب الخلط بين مخزون الترجمة TLB و مخزونات المعالج الأخرى L1 & L2 caches. رغم أن مبدأ المحلية يسري على جميع هذه المخزونات.

ينبغي جيدا إدراك أن الأخذ بعين الاعتبار عملية التخزين في TLB ليس ترفا أو اختيارا يمكن تجاهله. تذكر جيدا أنه إذا تم تغيير الترجمة لصفحة ما – مثلا بفهرسة صورة صفحة في مدخل كان يشير سابقا إلى "غير موجود" - و لم يتم إخبار المعالج بهذا التغيير، فإن المعالج سيظل يستعمل الترجمة القديمة في حال ما كانت موجودة قبلا في مخزون الترجمة. لذلك ينبغي الانتباه إلى تحديث معلومات المعالج عند تغيير ترجمة كل مدخل سواء في مجلد الصفحات أو في جداول الصفحات.

لتحديث مداخل الTLB يمكننا استعمال التعليمة

<عنوان الصفحة> invlpg

سؤال كيف اخبر المعالج بأنه يجب ان ينتقل إلي نمط الذاكرة الوهميه وكيه اخبره بمكان الجدول العام ؟

اخبار المعالج بمكان الجدول عن طريق تحميل عنوانه إلي CR3
الانتقال إلي نمط الذاكرة الوهميه عن طريق جعل البت PG = 1 وهو كالأتي
Mov eax,Cr0
Or eax,0x80000000
Mov eax,Cr0

لنأتي الآن إلي الجزء العملي
سوف نطلق علي الجدول العلم directory أي مجلد
والجدول الرئيس table ومنطقه المعلومات page أي صفحه
قم بعمل ملف جديد وسمه vmem.h وقم بكتابه الكود التالي

Vmem.h

```
#ifndef _H_virtual_memory_
#define _H_virtual_memory_
#include "phmem.h"
#include "system.h"
class vmem
{
    uint baseDeneid,topDeneid;
public:
    void install(uint kbase,uint ktop);
    void alloc(uint vadd,uint padd,uint add);
    void free(uint vadd);
};
#endif
```

هذا لابد وانه مفهوم

انشئ ملف جديد وسمه vmem.cpp

Vmem.cpp

```

inline void makePde(uint *pde,uint paddr,uint flags)
{*pde = paddr | (flags & (4096 - 1));}

inline void makePte(uint *pte,uint paddr,uint flags)
{*pte = paddr | (flags & (4096 - 1));}

inline uint pdeIndex(uint vaddr)
{return (vaddr >> 22);}

inline uint* pteAddr(uint vaddr)
{ return (uint*)(255<<22)+(vaddr>>12);}

inline uint* pdeAddr(uint vaddr)
{return ((uint*)((255<<22)+(255<<12)))+pdeIndex(vaddr));}

inline uint pteIndex(uint vaddr)
{return ((vaddr >> 12)& 0x3ff);}

uint *kpde;uint *kpte;

/*****/
void vmem::install(uint kbase,uint ktop)
{
    bothmem phm;

    uint *pd = (uint*)phm.allocpg();
    for (uint paddr = kbase; paddr < ktop; paddr+=4096)
    {
        kpde = pd + pdeIndex(paddr);
        if(!(*kpde & 1)) {

            kpte = (uint *)phm.allocpg();
            makePde(kpde, (uint)kpte, 1|2);

        }
        makePte(kpte+pteIndex(paddr), paddr, 1|2);
    }
    makePde(pd+255, (uint)pd, 1|2);
    uint pbdr = (uint)pd;
    asm volatile (
        "movl %0, %%cr3    \n\t\
        movl %%cr0, %%eax \n\t\
        orl $0x80000000, %%eax \n\t\
        movl %%eax, %%cr0    \n\t\
        jmp    1f    \n\t\
        1: \n\t\
        ::"r"(pbdr):"memory","eax");
    baseDeneid = kbase;
    topDeneid = ktop;
}

void vmem::alloc(uint vadd,uint padd,uint add)

```

```

{
    bothmem phm;
    padd = (padd & ~(4096-1));
    vadd = (vadd & ~(4096-1));
    if(isSystemAddr(vadd) || isMapAddr(vadd))return ;
    uint *pde = pdeAddr(vadd);
    if(!(*pde & 1))
    {
        uint *pt = (uint*)((255<<22)+(pdeIndex(vadd)*4096));
        uint ppt = phm.allocpg();

        if(!ppt)return ;

        makePde(pde, (uint)ppt, add);
        asm volatile("invlpg %0" ::"m"(*(uint *)pt));
        for (int i = 0; i < 1024; ++i) {pt[i] = 0;}

    }
    uint *pte = pteAddr(vadd);
    makePte(pte, (uint)padd,add);
    asm volatile("invlpg %0" ::"m"(*(uint *)vadd));
}
void vmem::free(uint vadd)
{
    vadd = (vadd & ~(4096-1));
    if(isSystemAddr(vadd) || isMapAddr(vadd))return;
    uint *pde = pdeAddr(vadd);
    if((*pde & 1)) {
        *(pteAddr(vadd)) = 0;
    }
    return;
}
}

```

جرب الان ان تكمل وحدك 😊 واي اسئله في المنتدى او علي البريد الالكتروني