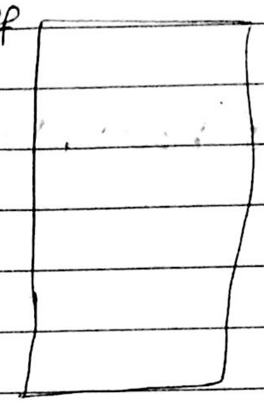
Section
27/11/2014

Phase 2: Memory management

① 4Gb ff لو هحمل load للبرنامج الـ Kernel أو اي برنامج في الـ Memory له البرنامج مكتوب الـ memory في مساحة معينة 3G ← 3.5G
واذا الميموري كانت عندي 1G فكده البرنامج هيضرب # أول مشكلة

physical memory



② لو برنامجين بيستعملوا نفس المكان في الميموري وكل واحد حط data على التانية overwrite فكده الـ data هترجع للبرنامج غلط.

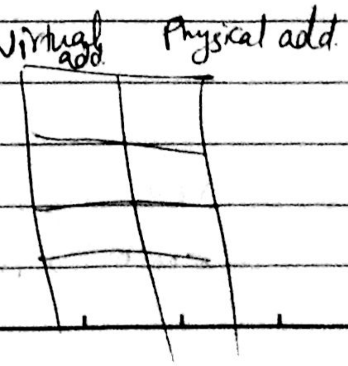
③ لو كان عندي برامج اخدت مساحات من الميموري وخلصت ولسه فيه برامج عايزة تشتغل

حل المشاكل السابقة هو: virtual memory
هنتخيل انه فيه memory واخده مساحة مثلاً 3G أو 4Gb

كده لو عندي برنامج هيستعمل مثلاً address 0X124 هنعمل mapping مباشر الـ address دا وعمل له map في مكان في الـ virtual mem. ولما برنامج تاني عايز يستعمل نفس الـ address نعمل map للعنوان دا انه في الـ virtual mem بتاعه بـ 0X10 مثلاً فكده البرنامج مش حاسس بالفرق بكم كده ضبطت الميموري

كذلك بستعمل الـ map لو البرنامج عايز من 3G ← 3.5G مستعمل الـ map علشان يخلي الـ address يبقى في حدود الـ 1G بتاعت الميموري بتاعتي.

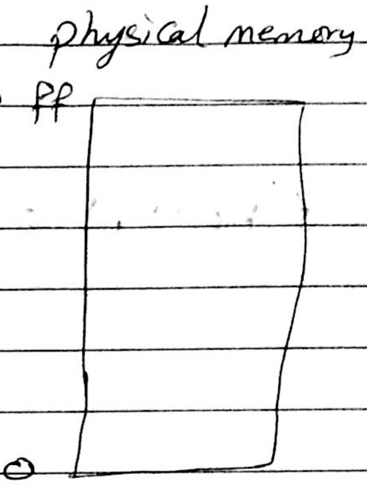
* Mapping is applied using table.



Mapping table

Section
27/11/2014

Phase 2: Memory management



① لو هحمل load للبرنامج Kernel أو اي برنامج
في ال Memory لو البرنامج مكتوب انه هيحط
في مساحة من 3G ← 3.5G
وانا المموري كانت عندي 1G فكده
البرنامج هيضرب # أول مشكلة

② لو برامج بيستعملوا نفس المكان في المموري وكل واحد حط داتا عملت
overwrite للتانية فكده الداتا هترجع للبرنامج غلط.

③ لو كان عندي برامج اخدت مساحات من المموري وخلصت ولسه فيه برامج عايزة تشتغل

حل المشاكل السابقة هو: virtual memory
هو متخيل انه فيه memory واخده مساحة مثلاً اد ال 4Gb

كده لو عندي برنامج هيستعمل مثلاً address 0x124 هعمل mapping هاخد
ال address دا وهعمله map في مكانه في ال virtual mem. ولو برنامج تاني جه
يستعمل نفس ال address نعمل map العنوان دا انه في ال ~~physical~~ virtual mem بتاعه
ل 0x10 مثلاً فكده البرنامج مش حاسس بالفرق لكن كده ضبطت المموري

كذلك بستعمل ال map لو البرنامج عايز من 3G ← 3.5G هستعمل ال map
عشان يخلي ال address يبقى في حدود ال 1G بتاعت المموري بتاعتي.

* Mapping is upplied using table.

Virtual add	Physical add

Mapping table

ال table حطيناها في ال RAM (Main Mem)

← اي برنامج بنعمله load من ال disc لل Memory و هو لو كانت عليا بيتشيل مرة منها اللي مش مستعمل بتحطه في ال disc وتكون مكانه البرنامج

← هنقسم ال Memory إلى pages هيقسم ال Mem. إلى blocks كل واحد 4 kb وكل 1Gb تتقسم إلى 1Ks (0 → 4095) one page

وكذلك ال virtual هتتقسم إلى pages.

ونعمل map لل addresses ع ال pages

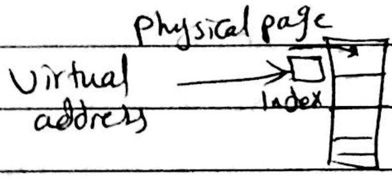
← كل برنامج له ال table الخاص به لأن كل برنامج شغال على virtual mem خاص به

* address:-

address in virtual add. 32 bit

ex, add: 0x0000 0003

عشان أعمل له map في ال physical address باخد ال الجزء دا وبشوف في الجدول ال 0 وجاي في ال main رقم كام مثلاً لو طلع 02 هنبدل ال 0000 في ال address إلى 0X0002 ، وال 0003 ال offset اللي هيبدأ منه في ال page



← الجدول عكس ميبقاش 2 columns عكم يكون

يبقى ال index يمثل ال virtual address والمحتوى هو ال physical address المقابل له

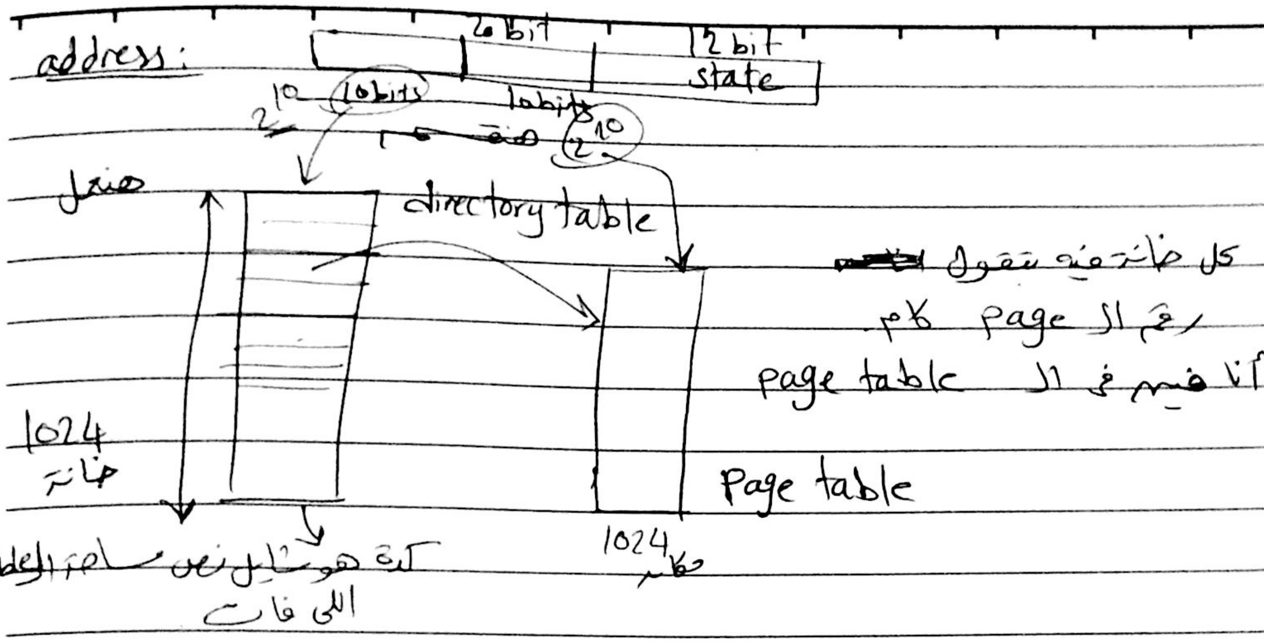
هنلاقي فيه 12 bit زيادة في ال physical address دول اللي بيحددوا حالة ال page سواء موجودة ولا blocked ولا clean or dirty

(State,)

* ال table عكم يبقى كبير فعشان نقسم الموضوع دا بنعمل multilevel

(يأخذ وقت كبير لتحميل الجدول وال check عليه)

الKernel هتعمل كدة = احنا اللى هنعمل OS



* page table: دا اللى فيه رقم ال physical add. المقابل لل virtualmem
كدة كل directory table له عدد page table = 1024
كدة أسرع لأنه بدل ما كل مرة يحمل table كله 1024x1024 دلوقتى فقط يحمل 1024 x عدد ال tables اللى عايزاهم فقط.

← حل تانى للتقسيم: look aside
عشان ال Mem. بطيئة فبدل ما أطلع ال address من ال page table فى الميموري ضيفنا buffer (HW) حطينا فيه ال page table فنعمل ال Map مباشرة معاه أسرع من الميموري وبيستغرق أقل من clk cycle

تسليم phase 2 : بعد أسبوعين