‬ به نام خدا



دانشکده‌های فنی

دانشکده برق و کامپیوتر

دانشگاه تهران

**سیستم عامل**

**گزارش آزمایش4 - آزمایشگاه سیستم عامل**

**هم گام سازی**

**گروه ۱‌**

**علیرضا زارع نژاد**

**محمدضا عظیمی**

**امیرحسین عبادی**

****

**سوالات:**

**علت غیرفعال کردن وقفه چیست‌؟ توابع pushcli , popcli به چه منظوری استفاده شده و چه تفاوتی با cli , sti دارند‌؟**

غیر فعال کردن interrupt ها به منظور حفاظت از critical section انجام می شود.

تابع cli وقفه های سیستم عامل را disable می کند و در مقابل sti وقفه ها را enable می کند. تابع pushcli خود از تابع cli , تابع popcli نیز خود از sti استفاده می کند.

درون struct cpu یک متغیر int وجود دارد به نام ncli که در صورت صفر بودن به معنی فعال بودن وقفه ها و در صورتی که صفر نباشد به معنای غیر فعال بودن وقفه ها است . حال اگر pushcli فراخوانی شود و وقفه فعال بوده باشد وقفه را غیر فعال می کند با فراخوانی کردن تابع cli. و به مقدار ncli یک واحد می افزاید. اگر قبل از فراخوانی این تابع وقفه غیر فعال بوده باشد صرفا یک واحد به متغیر اضافه می شود.

در مقابل با فراخوانی popcli مقدار متغیر یک واحد کاهش می یابد و در صورتی که مقدار متغیر ncli صفر شده باشد از حالت غیرفعال خارج می شود و وقفه فعال می شود.( با تابع sti )

**حالات مختف پردازه ها در xv6 . تابع sched چه کاربردی دارد؟**

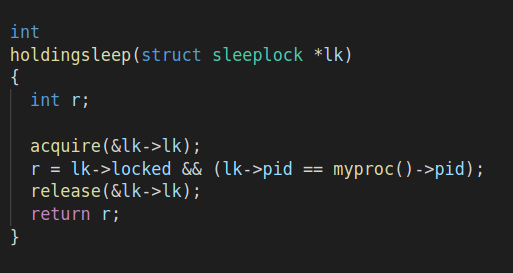
تابع sched این امکان را می دهد که بین پردازه ها switch کنیم بدین شکل که قبل از switch کردن با فراخوانی تابع swtch و context پردازه فعلی را به schedular می دهد تا در proc→context ذخیره کند و state پردازه بعدی را هم runnable می کند و به پردازه جدید switch می کند.

حالت های مختلف پردازه در xv6:

* **unused‌** : تمامی پردازه ها به صورت پیش فرض در این حالت قرار دارند . یعنی اگر پردازه ای دستور fork را اجرا کند اولین پردازه ای که در این حالت قرار دارد انتخاب شده و پس از تغییر حات اجرا می شود.
* **EBRIO‌ :** وضعتی است که پردازه از حالت unused خارج شده و یک pid مخصوص به خود می گیرد و در حال initialize شدن است.
* **Sleeping‌ :** به حالتی گفته می شود که در آن پردازه هیچ دستورالعملی را اجرا نمی کند و CPU را رها کرده باشد
* **Runnable :** به حاتی گفته می شود که در آن پردازه آماده به اجرا شدن است و منتظر است که CPU را در اختیار بگیرد.
* **Running‌ :** هنگامی که پردازه در حال اجرای دستور العمل باشد و CPU در اختیارش باشد

* **Zombie ‌:** به حالتی گفته می شود که پردازه در terminate state قرار دارد. این حالت زمانی پیش می آید که ما از فراخوانی سیستمی fork استفاده می کنیم و یک child را ایجاد می کنیم . هنگامی که بچه دستور العمل هایش را به طور کامل اجرا کرد exit status خود را به پدر خود می فرستد . تا زمانی که پدر این پیام را از بچه خود دریافت کند بچه در حالت زامبی قرار دارد. یعنی اجرا شده ولی از ptable خارج نمی شود.

**می توان با اعمال تغییری در توابع دسته دوم امکان آزادسازی را تنها برای پردازه صاحب قفل میسر نمود. قفل معادل در هسته لینوکس را به طور مختصر معرفی نمایید.**

****

**‬‬**

در فایل sleeplock.c تابع releasesleep را به گونه ای تغییر می دهیم تا تنها صاحب قفل قادر به آزاد کردن آن باشد . برای این کار یک تابع جدید به نام holdingsleep را اضافه می کنیم ه پیاده سازی آن به صورت زیر است.

در این تابع چک می کنیم که اگر قفلی که سعی در آزاد کردن آن داریم قفل شده است یا خیر. اگر قفل شده است شناسه ی پردازه ای که قصد دارد آن را آزاد کند ( پردازه در حال اجرا‌) همان شناسه ای است که در struct sleeplock ذخیره شده است یعنی شناسه همان پردازه ای که قبلا آن را گرفته است ) مقدار بازگشتی این تابع در صورتی یک خواهد بود که صاحب قفل قصد آزاد کردن آن را داشته باشد

قفل معادل sleeplock در هسته ی لینوکس semaphore است که در لینک زیر آورده شده است.

<https://github.com/torvalds/linux/blob/6f0d349d922ba44e4348a17a78ea51b7135965b1/include/linux/semaphore.h>

<https://github.com/torvalds/linux/blob/6f0d349d922ba44e4348a17a78ea51b7135965b1/kernel/locking/semaphore.c>

**پیاده سازی ماکروی barrier در لینوکس در معماری x86:**

***/\* SPDX-License-Identifier: GPL-2.0 \*/***

#ifndef [**\_\_ASM\_BARRIER\_H**](https://elixir.bootlin.com/linux/latest/ident/__ASM_BARRIER_H)

#define [**\_\_ASM\_BARRIER\_H**](https://elixir.bootlin.com/linux/latest/ident/__ASM_BARRIER_H)

#ifndef \_\_ASSEMBLY\_\_

#define [**nop**](https://elixir.bootlin.com/linux/latest/ident/nop)() [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_("mov\tr0,r0\t@ nop\n\t");

#if \_\_LINUX\_ARM\_ARCH\_\_ >= 7 || \

(\_\_LINUX\_ARM\_ARCH\_\_ == 6 && [**defined**](https://elixir.bootlin.com/linux/latest/ident/defined)(CONFIG\_CPU\_32v6K))

#define [**sev**](https://elixir.bootlin.com/linux/latest/ident/sev)() [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("sev" : : : "memory")

#define [**wfe**](https://elixir.bootlin.com/linux/latest/ident/wfe)() [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("wfe" : : : "memory")

#define [**wfi**](https://elixir.bootlin.com/linux/latest/ident/wfi)() [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("wfi" : : : "memory")

#else

#define [**wfe**](https://elixir.bootlin.com/linux/latest/ident/wfe)() do { } while (0)

#endif

#if \_\_LINUX\_ARM\_ARCH\_\_ >= 7

#define [**isb**](https://elixir.bootlin.com/linux/latest/ident/isb)([**option**](https://elixir.bootlin.com/linux/latest/ident/option)) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("isb " #[**option**](https://elixir.bootlin.com/linux/latest/ident/option) : : : "memory")

#define [**dsb**](https://elixir.bootlin.com/linux/latest/ident/dsb)([**option**](https://elixir.bootlin.com/linux/latest/ident/option)) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("dsb " #[**option**](https://elixir.bootlin.com/linux/latest/ident/option) : : : "memory")

#define [**dmb**](https://elixir.bootlin.com/linux/latest/ident/dmb)([**option**](https://elixir.bootlin.com/linux/latest/ident/option)) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("dmb " #[**option**](https://elixir.bootlin.com/linux/latest/ident/option) : : : "memory")

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_THUMB2\_KERNEL

#define [**CSDB**](https://elixir.bootlin.com/linux/latest/ident/CSDB) ".inst.w 0xf3af8014"

#else

#define [**CSDB**](https://elixir.bootlin.com/linux/latest/ident/CSDB) ".inst 0xe320f014"

#endif

#define [**csdb**](https://elixir.bootlin.com/linux/latest/ident/csdb)() [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_([**CSDB**](https://elixir.bootlin.com/linux/latest/ident/CSDB) : : : "memory")

#elif [**defined**](https://elixir.bootlin.com/linux/latest/ident/defined)(CONFIG\_CPU\_XSC3) || \_\_LINUX\_ARM\_ARCH\_\_ == 6

#define [**isb**](https://elixir.bootlin.com/linux/latest/ident/isb)(x) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("mcr p15, 0, %0, c7, c5, 4" \

: : "r" (0) : "memory")

#define [**dsb**](https://elixir.bootlin.com/linux/latest/ident/dsb)(x) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("mcr p15, 0, %0, c7, c10, 4" \

: : "r" (0) : "memory")

#define [**dmb**](https://elixir.bootlin.com/linux/latest/ident/dmb)(x) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("mcr p15, 0, %0, c7, c10, 5" \

: : "r" (0) : "memory")

#elif [**defined**](https://elixir.bootlin.com/linux/latest/ident/defined)(CONFIG\_CPU\_FA526)

#define [**isb**](https://elixir.bootlin.com/linux/latest/ident/isb)(x) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("mcr p15, 0, %0, c7, c5, 4" \

: : "r" (0) : "memory")

#define [**dsb**](https://elixir.bootlin.com/linux/latest/ident/dsb)(x) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("mcr p15, 0, %0, c7, c10, 4" \

: : "r" (0) : "memory")

#define [**dmb**](https://elixir.bootlin.com/linux/latest/ident/dmb)(x) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("" : : : "memory")

#else

#define [**isb**](https://elixir.bootlin.com/linux/latest/ident/isb)(x) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("" : : : "memory")

#define [**dsb**](https://elixir.bootlin.com/linux/latest/ident/dsb)(x) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("mcr p15, 0, %0, c7, c10, 4" \

: : "r" (0) : "memory")

#define [**dmb**](https://elixir.bootlin.com/linux/latest/ident/dmb)(x) [**\_\_asm\_\_**](https://elixir.bootlin.com/linux/latest/ident/__asm__) \_\_volatile\_\_ ("" : : : "memory")

#endif

#ifndef [**CSDB**](https://elixir.bootlin.com/linux/latest/ident/CSDB)

#define [**CSDB**](https://elixir.bootlin.com/linux/latest/ident/CSDB)

#endif

#ifndef [**csdb**](https://elixir.bootlin.com/linux/latest/ident/csdb)

#define [**csdb**](https://elixir.bootlin.com/linux/latest/ident/csdb)()

#endif

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_ARM\_HEAVY\_MB

extern void (\*[**soc\_mb**](https://elixir.bootlin.com/linux/latest/ident/soc_mb))(void);

extern void [**arm\_heavy\_mb**](https://elixir.bootlin.com/linux/latest/ident/arm_heavy_mb)(void);

#define [**\_\_arm\_heavy\_mb**](https://elixir.bootlin.com/linux/latest/ident/__arm_heavy_mb)(x...) do { [**dsb**](https://elixir.bootlin.com/linux/latest/ident/dsb)(x); [**arm\_heavy\_mb**](https://elixir.bootlin.com/linux/latest/ident/arm_heavy_mb)(); } while (0)

#else

#define [**\_\_arm\_heavy\_mb**](https://elixir.bootlin.com/linux/latest/ident/__arm_heavy_mb)(x...) [**dsb**](https://elixir.bootlin.com/linux/latest/ident/dsb)(x)

#endif

#if [**defined**](https://elixir.bootlin.com/linux/latest/ident/defined)(CONFIG\_ARM\_DMA\_MEM\_BUFFERABLE) || [**defined**](https://elixir.bootlin.com/linux/latest/ident/defined)([**CONFIG\_SMP**](https://elixir.bootlin.com/linux/latest/ident/CONFIG_SMP))

#define [**mb**](https://elixir.bootlin.com/linux/latest/ident/mb)() [**\_\_arm\_heavy\_mb**](https://elixir.bootlin.com/linux/latest/ident/__arm_heavy_mb)()

#define [**rmb**](https://elixir.bootlin.com/linux/latest/ident/rmb)() [**dsb**](https://elixir.bootlin.com/linux/latest/ident/dsb)()

#define [**wmb**](https://elixir.bootlin.com/linux/latest/ident/wmb)() [**\_\_arm\_heavy\_mb**](https://elixir.bootlin.com/linux/latest/ident/__arm_heavy_mb)([**st**](https://elixir.bootlin.com/linux/latest/ident/st))

#define [**dma\_rmb**](https://elixir.bootlin.com/linux/latest/ident/dma_rmb)() [**dmb**](https://elixir.bootlin.com/linux/latest/ident/dmb)(osh)

#define [**dma\_wmb**](https://elixir.bootlin.com/linux/latest/ident/dma_wmb)() [**dmb**](https://elixir.bootlin.com/linux/latest/ident/dmb)(oshst)

#else

#define [**mb**](https://elixir.bootlin.com/linux/latest/ident/mb)() [**barrier**](https://elixir.bootlin.com/linux/latest/ident/barrier)()

#define [**rmb**](https://elixir.bootlin.com/linux/latest/ident/rmb)() [**barrier**](https://elixir.bootlin.com/linux/latest/ident/barrier)()

#define [**wmb**](https://elixir.bootlin.com/linux/latest/ident/wmb)() [**barrier**](https://elixir.bootlin.com/linux/latest/ident/barrier)()

#define [**dma\_rmb**](https://elixir.bootlin.com/linux/latest/ident/dma_rmb)() [**barrier**](https://elixir.bootlin.com/linux/latest/ident/barrier)()

#define [**dma\_wmb**](https://elixir.bootlin.com/linux/latest/ident/dma_wmb)() [**barrier**](https://elixir.bootlin.com/linux/latest/ident/barrier)()

#endif

#define [**\_\_smp\_mb**](https://elixir.bootlin.com/linux/latest/ident/__smp_mb)() [**dmb**](https://elixir.bootlin.com/linux/latest/ident/dmb)(ish)

#define [**\_\_smp\_rmb**](https://elixir.bootlin.com/linux/latest/ident/__smp_rmb)() [**\_\_smp\_mb**](https://elixir.bootlin.com/linux/latest/ident/__smp_mb)()

#define [**\_\_smp\_wmb**](https://elixir.bootlin.com/linux/latest/ident/__smp_wmb)() [**dmb**](https://elixir.bootlin.com/linux/latest/ident/dmb)(ishst)

#[**ifdef**](https://elixir.bootlin.com/linux/latest/ident/ifdef) CONFIG\_CPU\_SPECTRE

static inline unsigned long [**array\_index\_mask\_nospec**](https://elixir.bootlin.com/linux/latest/ident/array_index_mask_nospec)(unsigned long idx,

unsigned long sz)

{

unsigned long mask;

[**asm**](https://elixir.bootlin.com/linux/latest/ident/asm) [**volatile**](https://elixir.bootlin.com/linux/latest/ident/volatile)(

"cmp %1, %2\n"

" sbc %0, %1, %1\n"

[**CSDB**](https://elixir.bootlin.com/linux/latest/ident/CSDB)

: "=r" (mask)

: "r" (idx), "Ir" (sz)

: "cc");

[**return**](https://elixir.bootlin.com/linux/latest/ident/return) mask;

}

#define [**array\_index\_mask\_nospec**](https://elixir.bootlin.com/linux/latest/ident/array_index_mask_nospec) [**array\_index\_mask\_nospec**](https://elixir.bootlin.com/linux/latest/ident/array_index_mask_nospec)

#endif

#include <[**asm-generic/barrier.h**](https://elixir.bootlin.com/linux/latest/source/include/asm-generic/barrier.h)>

#endif */\* !\_\_ASSEMBLY\_\_ \*/*

#endif */\* \_\_ASM\_BARRIER\_H \*/*

**هم چنین برای توضیحات بیشتر لینک های زیر آورده شده است.**

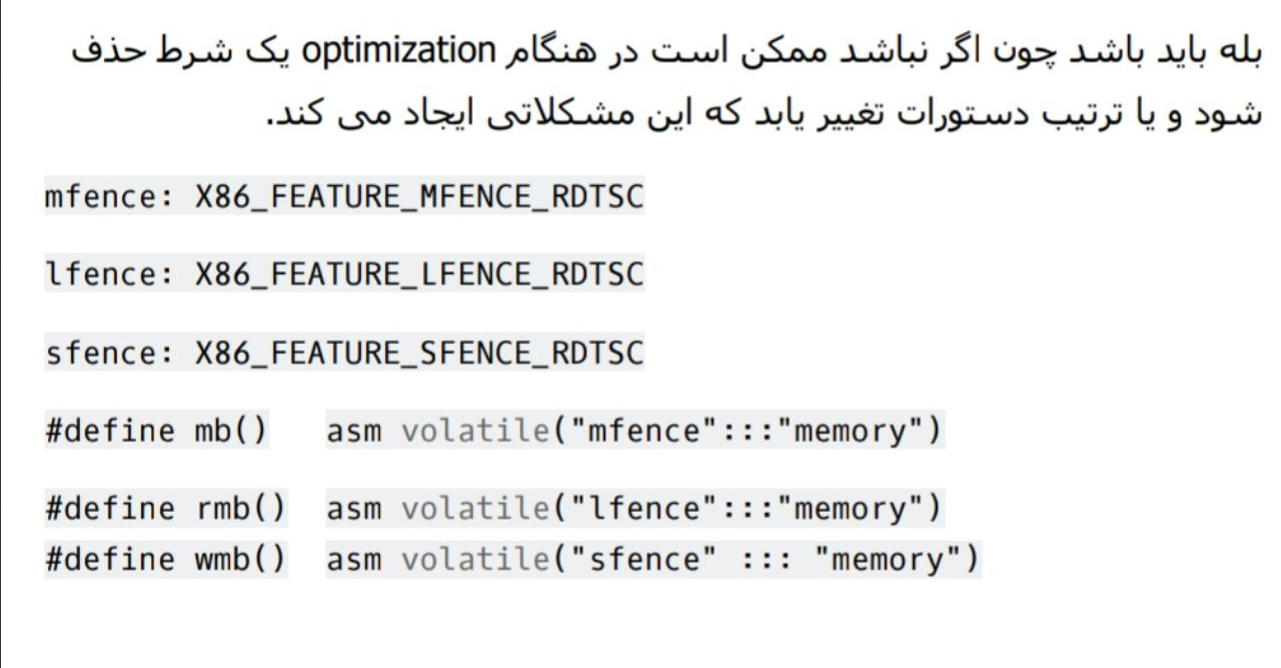
[**https://elixir.bootlin.com/linux/latest/source/tools/include/asm-generic/barrier.h**](https://elixir.bootlin.com/linux/latest/source/tools/include/asm-generic/barrier.h)

[**https://elixir.bootlin.com/linux/latest/source/tools/include/asm/barrier.h**](https://elixir.bootlin.com/linux/latest/source/tools/include/asm/barrier.h)

[**https://elixir.bootlin.com/linux/latest/source/include/asm-generic/barrier.h**](https://elixir.bootlin.com/linux/latest/source/include/asm-generic/barrier.h)

[**https://elixir.bootlin.com/linux/latest/source/arch/x86/include/asm/barrier.h**](https://elixir.bootlin.com/linux/latest/source/arch/x86/include/asm/barrier.h)

**آیا یک دستور مانع حافظه باید مانع بهینه سازی هم باشد؟ نام ماکروی پیاده سازی سه نوع مانع حافظه در لینوکس در معماری x86 را به همراه دساورالعمل های ماشین پیاده سازی آن ها را ذکر کنید .**



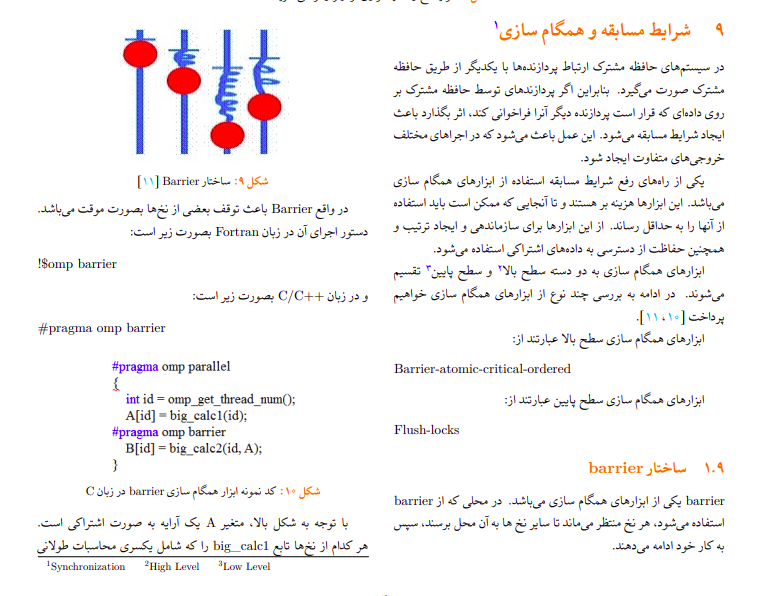
**یک کاربرد از مانع در پرادازش موازی را بنویسید؟**

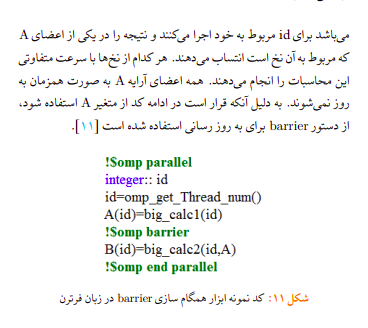
[**https://en.wikipedia.org/wiki/Barrier\_(computer\_science**](https://en.wikipedia.org/wiki/Barrier_(computer_science)[**)**](https://en.wikipedia.org/wiki/Barrier_(computer_science))

در لینک بالا توضیح مختصر و مفیدی در رابطه با کاربرد بریر در هماهنگ سازی و به طور بهتر همگام سازی پراسس ها آورده شده است.

در یک مثال کلی می توان به هماهنگ کردن صدا و تصویر اشاره کرد.

پردازش موازی اجرای یک [فرایند](https://fa.wikipedia.org/wiki/فرایند) به طور هم‌زمان، عموماً با تقسیم عملیات پردازش بر روی چندین [پردازنده](https://fa.wikipedia.org/wiki/پردازنده) به منظور افزایش کارایی و در نهایت سرعت بخشیدن برای رسیدن به جواب است. گاهی استفاده از تکنیکهای اشتراک زمان را در یک پردازنده، به اشتباه پردازش موازی به حساب می‌آورند (چند پروسه به طور موازی روی یک پردازنده اجرا می‌شوند). ایدهٔ این کار بر این مبنا است که هر مسئله به طور معمول قابل تقسیم به چندین مسئله با اندازهٔ کوچک‌تر است که این مسئله‌های کوچک‌تر می‌توانند به صورت هم‌زمان [حل شده](https://fa.wikipedia.org/wiki/حل_شده) و در نهایت ادغام شوند تا نتیجه نهایی سریع‌تر بدست آید.





هنگامی که تردها سرعت های متفاوتی دارند و انجام صحیح یک پراسس نیازمند به همه ی آن ها خواهد بود که در اینجا سینک شدن کاملا قابل درک خواهد بود.