باسمه تعالی

فاز شماره یک درس سیستم عامل – استاد جوادی

سید امیرمهدی میرشریفی – ۹۸۳۱۱۰۵

۱) یک توضیح کلی از فرایند های طی شده برای ساخته شدن اولین process ارائه دهید. سعی داشته باشید که این مراحل را با ارجاع دادن به کدها و اسکرین شات کامل کنید .

در ابتدا و بعد از آن که هسته سیستم عامل به حافظه منتقل می شود ، پردازنده سیستم عامل را با فراخوانی فایل entry.S اجرا می کند.
این فایل اسمبلی یک استک را برای سیستم عامل اجرا می کند تا سیستم عامل بتواند فایل های c را اجرا کند.

```
_entry:
        # set up a stack for C.
        # stack0 is declared in start.c,
        # with a 4096-byte stack per CPU.
        # sp = stack0 + (hartid * 4096)
        la sp, stack0
        li a0, 1024*4
        csrr a1, mhartid
        addi a1, a1, 1
        mul a0, a0, a1
        add sp, sp, a0
        # jump to start() in start.c
        call start
```

همان طور که در تصویر مشاهده می کنید در پایان تابع start.c فراخوانی می شود .

تابع start برخی تنظیمات را انجام می دهد و این در حالی است که انجام آن تنظیمات صرفا در machine mode امکان پذیر است و در حالتsupervisor امکان پذیر است.

بنابراین این تابع پس از پایان کار ، با استفاده از تابع mret به حالت supervisor mode بر می گردیم. با توجه به این که در سیستم عامل xv6-riscv ۴ حالت برای دستور های privilege وجود دارد، مطلب پایین که در ادامه آورده شده است جهت آشنایی بیشتر و توضیح machine mode & supervisor mode که به آن اشاره شد ، است.

RISC-V features four privilege levels: Machine mode, hypervisor mode, supervisor mode, and user mode. Only machine mode is mandatory. Machine mode has access to all the hardware features but does not have virtual-memory support. Hypervisor mode is meant to be used for virtualization.
As of the time of writing, the hypervisor mode ISA has not been specified. Supervisor mode is the level where an operating-system kernel is supposed to be executed. In contrast to the machine mode, this mode implements the MMU and offers a variety of page-table formats.User mode is - as usual - the place where user-level code is executed.

تابع start ، چیزی به عنوان خروجی بر نمی گرداند و صرفا برخی از تنظیمات را انجام می دهد که میتوان به:

تغییر محتوای رجیستر حالت به supervisor mode

```
// set M Previous Privilege mode to Supervisor, for mret.
unsigned long x = r_mstatus();
x &= ~MSTATUS_MPP_MASK;
x |= MSTATUS_MPP_S;
w_mstatus(x);
```

واگذار کردن وقفه ها و استثناء ها به supervisor mode

```
// delegate all interrupts and exceptions to supervisor mode.
w_medeleg( x: 0xffff);
w_mideleg( x: 0xffff);
w_sie( x: r_sie() | SIE_SEIE | SIE_STIE | SIE_SSIE);
```

و نوشتن آدرس تابع main در رجیستر mepc به عنوان مقصد بعدی

```
// set M Exception Program Counter to main, for mret.
// requires gcc -mcmodel=medany
w_mepc( x: (uint64)main);
```

و تنظیم چیپ ساعت و تولید شمارشگر وقفه ، اشاره نمود.

پس از بازگشت تابع به main و شناسایی چندین دیوایس و سیستم اولین پردازه با صدا زده شدن تابع userinit ساخته می شود.

Userinit یک برنامه کوچک را اجرا میکند که در یک فایل اسمبلی نوشته شده است و وظیفه آن به وجود آوردن اولین سیستم کال است.

۲) تحلیل کنید که مقدار برگردانده شده نزدیک مقدار نظری از پیش تعیین شده است یا خیر.

در فایل memlayout.c و با توجه به خط کدی که در ادامه می بینید میشود ظرفیتی که به سیستم اختصاص یافته است را به دست آورد که این عدد ۱۳۴۲۱۷۷۲۸ بایت است

```
#define KERNBASE 0x80000000L
#define PHYSTOP (KERNBASE + 128*1024*1024)
```

در ادامه و با استفاده از سیستم کال که در این فاز پروژه طراحی شد، قبل از به وجود آمدن پردازه کاربر ، استعلام حافظه را می گیریم و عددی که بر میگردد برابر ۱۳۴۰۷۸۴۶۴ بایت است. همانطور که مشاهده می کنید این اعداد یکسان نیستند و ۱۳۹,۲۶۴ بایت تفاوت دارند.

توضیحات را می توانید در عکس زیر مشاهده بکنید.

```
xv6 kernel is booting

xv6 free memory before user processes start: 134078464 byte.
hart 2 starting
hart 1 starting
init: starting sh
$ kfreemem
---------------------------------
xv6 system memory: 134217728 byte.
xv6 free   memory: 133382144 byte.
xv6 busy   memory: 835584    byte.
---------------------------------
```

۳) مراحل طی شده برای فراخوانی سیستم کال و برگردانده شدن نتایج را داخل سیستم عامل بررسی کنید.

در ادامه به بررسی مرحله به مرحله تولید سیستم کال ای که در این پروژه پیاده سازی شده است می پردازیم . همچنین این مراحل از لینک الگوبرداری شده است.

در ابتدا می بایست در فایل kalloc.c دو تابع kfreepages و kfreemem را پیاده سازی کرد.

در فایل kalloc.c توابعی که با حافظه در ارتباط هستند، پیاده سازی می شود.

تابع kfreepages تابعی است که صفحه های خالی حافظه (memory pages) را محاسبه می کند و عدد آن را به صورت یک عدد صحیح بر می گرداند.

سپس تابع kfreemem را پیاده سازی می کنیم. با توجه به آن که سایز هر صفحه یکسان است و هر کدام از صفحه ها ۴۰۹۶ بایت ظرفیت دارند ، تابع kfreemem ، تعداد صفحه های خالی ای که از تابع kfreepages به دست آمده است را ضرب در ظرفیت هر صفحه می کند و بر می گرداند.

در مرحله بعد ، می بایست پروتوتایپ توابعی که پیاده سازی کردیم را در defs.h اضافه کنیم.

حال که توابع مورد نیاز را پیاده سازی کردیم ، به سراغ ساخت سیستم کال مورد نیاز می رویم.

برای اضافه کردن یک سیستم کال ابتدا می بایست در هسته لینوکس و در فایل sysproc.c که فراخوانی های سیستمی در آن پیاده سازی شده اند ، سیستم کال مد نظر را پیاده سازی کرد.

بعد از آن که سیستم کال را پیاده سازی کردیم، می بایست نام آن را در فایل syscalls.h به عنوان یک سیستم کال اعلام و شماره متناظر با سیستم کالمان را تعریف کنیم. سپس در همین فایل تابعی که در sysproc.c پیاده سازی کردیم را به عنوان هندلر فراخوانیمان تعریف می کنیم.

در این مرحله تمام تنظیمات برای اضافه شدن سیستم کال مان در بخش هسته سیستم عامل تمام شده است و باید به دنبال تغییرات در بخش کد های کاربر باشیم.

در ادامه می بایست سیستم کالی که به هسته سیستم عامل اضافه کرده ایم را در فایل usys.pl که سیستم کال ها در آن برای بخش کاربر اضافه شده است ، اضافه می کنیم. همچنین پروتوتایپ تابعی که سیستم کال را فراخوانی می کند را در فایل user.h اضافه میکنیم.

حال نوبت اضافه کردن یه برنامه در سطح کاربر است که کاربر بتواند هنگام کار با سیستم عامل آن برنامه را اجرا کند. اگر این برنامه اجرا نشود و فقط تابعی که در سطح کاربر طراحی کردیم را در سیستم عامل فراخوانی بکنیم با پیام مشکل اجرا از طرف سیستم عامل روبرو خواهید شد.

پس از آن که برنامه را در بخش کد های کاربر اضافه کردید می بایست به آن قابلیت اجرا هم بدهید که می توانید با استفاده از تغییراتی در فایل Makefile به آن قابلیت اجرا بدهید.

اکنون کاربر با اجرای برنامه ای که نوشته واضافه شد سیستم کال مورد نظر را فراخوانی می کند .