باسمه تعالی

دانشگاه صنعتی اصفهان
سیستم های عامل_ تمرین دوم
موعد تحویل: جمعه ۷ آذر ۹۹

سوالات تئوری

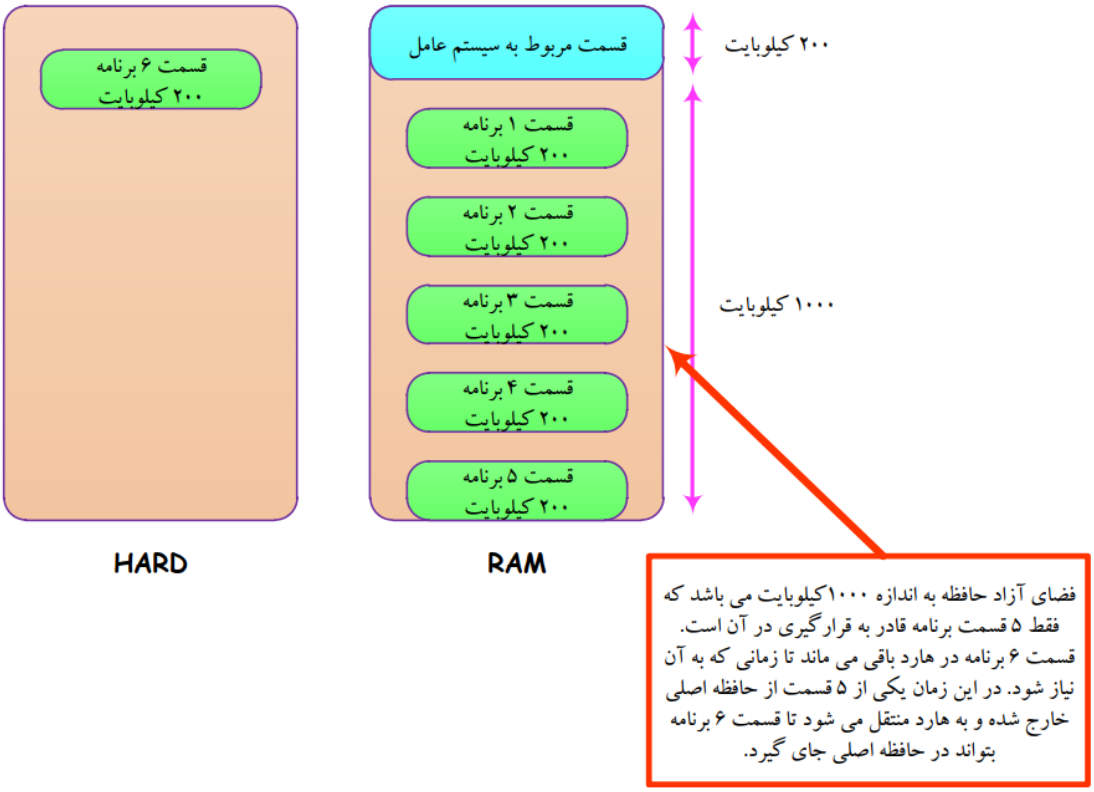
سوال ۱ (۵۰ نمره): در گذشته از روشی به نام تک برنامگی با قابلیت روی هم گذاری [۱] در مواردی که حافظه تخصیص داده شده به یک فرایند از اندازه فرایند کوچکتر بود، استفاده می شد. در این تکنیک، برنامه نویس برنامه ها را به قسمت هایی به همین نام تقسیم می کرد و تنها آن داده و دستورالعمل هایی در RAM قرار می گرفت که مورد نیاز بود. مابقی بخش ها در HARD باقی می ماند. هنگامی که به بخش های دیگری از آن برنامه نیاز باشد، قسمتی که مورد نیاز نیست از RAM خارج شده و قسمت مورد نیاز از HARD به RAM آورده می شود. روند اجرایی بدین صورت است که ابتدا Overlay0 اجرا شده و بعد از پایان اجرایش، Overlay1 را صدا میزند و به همین ترتیب ادامه می یابد. در سیستم هایی که امکان نگهداری چندین Overlay به طور همزمان در حافظه وجود داشت، سیستم عامل آنها را روی دیسک نگه می داشت و در مواقع لزوم، عملیات مبادله را انجام می داد. تقسیم برنامه به Overlay ها و فراخوانی آنها برعهده برنامه نویس بود و سیستم عامل فقط عمل مبادله را انجام می داد.

به طور مثال اگر برنامه ای به اندازه ۱۲۰۰ کیلوبایت باشد و حافظه آزاد نیز حجمی به اندازه ۱۰۰۰ کیلوبایت داشته باشد، می توان برنامه را به ۶ قسمت ۲۰۰ کیلوبایتی تقسیم کرد و ابتدا فقط ۵ قسمت یعنی ۱۰۰۰ کیلوبایت آن را داخل حافظه بار کرد. سپس در صورت نیاز به قسمت ۶ ام، یکی از قسمت هارا از حافظه اصلی خارج کرده و قسمت ۶ ام به داخل حافظه آورده شود.

شکل زیر شمایی را برای عملکرد این مثال ارائه می دهد.

---

[۱] Overlay



فرض کنید که قرار است برنامه ای در یک سیستم با مدیریت حافظه Overlay اجرا شود. این برنامه شامل زیر برنامه های [۲] مختلف است که با زیر برنامه A شروع و با خاتمه آن نیز پایان می پذیرد. در حین اجرای هر زیر برنامه، ممکن است که زیر برنامه دیگری نیز بعنوان بخش های دیگر برنامه اصلی مورد نیاز باشد. با توجه به اینکه اندازه فضای اولیه مورد نیاز برای زیر برنامه های مختلف بصورت زیر است:

A=5K, B=9K, C=10K, D=15K, E=7K, F=10K

الف) حداقل فضای مورد نیاز جهت اجرای برنامه اصلی به شیوه مدیریتی روی هم گذاری را تعیین نموده و ادعای خود را با دلیل توضیح دهید.

توجه: فراخوانی هر زیر برنامه نیاز به تخصیص فضای اولیه مورد نیاز آن خواهد داشت. سپس بر اساس تکنیک روی هم گذاری، هر زمان که نیاز به زیر برنامه ای باشد، می بایست فضای مورد نیاز آن نیز تخصیص داده شود. پایان یافتن هر زیر برنامه نیز به معنای عدم نیاز به حضور آن در حافظه اصلی خواهد بود.

[۲] Sub-Program(SP)

SP A;	SP B;	SP C;	SP D;	SP E;	SP F;
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
call SP D;	.	.	call SP E;	.	.
.	.	.	.	.	.
.	.	.	.	.	call SP C
.	.	.	.	.	.
call SP F;	.	.	call SP B;	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
end;	end;	end;	end;	end;	end;

ب) به طور مختصر بیان کنید که روش مدیریتی روی هم گذاری چه مزیت و مشکلی می تواند در بر داشته باشد؟

سوال ۲ (۶۰ نمره): کامپیوتری را با یک فضای آدرس پذیر مجازی ۶۴ بیتی که صفحات آن هر یک ۲۰۴۸ بایت ظرفیت دارند، در نظر بگیرید. اندازه هر مدخل جدول صفحه ۴ بایت است. به دلیل آنکه هر جدول باید داخل یک صفحه جای گیرد، یک جدول صفحه چند سطحی استفاده شده است.

الف) برای این جدول به چند سطح نیاز است؟ پاسخ خود را با رسم شماتیک حافظه مجازی و مشخص کردن ساختار بیت های آدرس برای قسمت های مختلف (آفست و بیت های مربوط به هر سطح جدول صفحه) نشان دهید.

ب) پاسخ سوال قسمت (الف) را برای حالتی که هر جدول صفحه ای معادل دو صفحه باشد، بیان کنید.

سوال ۳ (۶۰ نمره): حافظه اصلی کامپیوتری دارای چهار قاب صفحه است. زمان ورود، زمان آخرین دسترسی، بیت R(Reference)، بیت M(Modify) مربوط به هر یک از صفحات در جدول زیر آمده است. اگر خطای صفحه [۳] روی صفحه مجازی شماره ۴ در زمان ۳۱۹ رخ دهد. تحت الگوریتم های جایگزینی LRU و Clock محتویات کدام یک از قاب صفحه ها باید جابجا شوند؟

شماره صفحه مجازی	قاب صفحه	زمان ورود	زمان آخرین دسترسی	بیت M	بیت R
۲	۰	۱۲۵	۲۷۸	۱	۰
۱	۱	۲۲۹	۲۳۹	۰	۱
۰	۲	۱۱۹	۲۷۱	۰	۱
۳	۳	۱۵۹	۳۱۸	۱	۱

سوال ۴ (۶۰ نمره): اندازه صفحه در سیستمی با مدیریت حافظه مجازی و به صورت صفحه بندی درخواستی،

[۳] Page Fault (PF)

۲۵۶ بایت است. حافظه سیستم حاوی ۳ قاب صفحه (در ابتدا خالی) است. اجرای کد زیر در هر یک از حالات، منجر به چند نقص صفحه می شود؟

```
X: array[1 ... 128][1 ... 128] of byte
for register int=1 to 128 do
    for register int j=1 to 128 do
        X[i][j]=0;
```

الف) هر قاب صفحه می تواند به کد یا داده منتسب گردد و قاب های صفحه به اشتراک بین کد و داده استفاده می شوند. اندازه کد فرایند برابر یک صفحه است و فرض کنید که حافظه فرایند فقط از دو بخش کد و داده تشکیل می شود. اجرای کد را با استفاده از روش جایگزینی FCFS بررسی و تعداد خطای صفحه رخ داده را محاسبه نمایید.

ب) در این حالت، یکی از قاب های صفحه برای کد فرایند و دو قاب دیگر برای داده ها (آرایه) استفاده می شود. بررسی کنید که اگر از روش جایگزینی LRU استفاده گردد، اجرای این کد منجر به چند فقدان صفحه خواهد شد؟

توجه: در این سوال آرایه را به صورت ردیفی [۴] در حافظه در نظر بگیرید. هر عنصر آرایه نیز یک بایت است. بدیهی است که اندازه قاب صفحه را نیز باید ۲۵۶ بایت در نظر داشته باشید.

سوال ۵ (۵۰ نمره): در یک سیستم صفحه بندی، جدول صفحات در حافظه اصلی قرار دارد. زمان دسترسی به حافظه اصلی 50ns و زمان دسترسی به TLB برابر با 10ns و تعداد درایه های TLB برابر با 16 است. اگر در ابتدا درایه های TLB خالی باشد و فرایند در حال اجرا با اجرای دستورات دسترسی به داده، به ترتیب شماره صفحات آدرس مجازی 2 ,1, 3, 1, 2 را درخواست کند، درصد افزایش کارایی در هنگام استفاده از TLB چقدر خواهد بود؟

سوالات برنامه نویسی

سوال ۶ (۸۰ نمره):

الف) در رابطه با نحوه استفاده از `gdb` و `valgrind` و کاربرد آن‌ها تحقیق کنید و به طور مختصر توضیح دهید.

ب) برنامه ای بنویسید که با استفاده از `malloc` آرایه ای از اعداد integer بنام data با اندازه ۱۰۰ ایجاد نماید؛ سپس این آرایه را با مقدار صفر مقداردهی نمایید. با اجرای این برنامه چه اتفاقی می افتد؟ در صورتی که برنامه را با استفاده از `valgrind` اجرا نمایید چه نتیجه ای حاصل می شود؟ آیا برنامه صحیح است؟

[۴] Row-Major

ج) قسمت (ب) را با استفاده از فراخوانی سیستمی `mmap` انجام دهید و نتیجه حاصل را با نتیجه قسمت (ب) مقایسه کنید.

توضیح: در واقع نیاز است که بجای استفاده از `malloc` از فراخوانی سیستمی `mmap` استفاده نمایید.

ساختار کلی تابع بصورت زیر است:

```
void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);
```

برای اطلاعات بیشتر می‌توانید از دستور `man mmap` استفاده کنید.

د) برنامه ای بنویسید که با استفاده از mmap آرایه‌ای از اعداد integer به طول ۵ ایجاد کند و آن ها را با ۱،۲،۳،۴،۵ مقداردهی اولیه کنید. پس از مقدار دهی اولیه یک پروسس جدید ایجاد کنید. وظایف هر کدام از پروسس ها مقابل آن ها نوشته شده است:

- **پروسس فرزند:** مقادیر موجود در آرایه را ۱۰ برابر می‌کند.
- **پروسس والد:** پس از پایان پروسس فرزند مقادیر موجود در آرایه را چاپ می‌کند.

(این برنامه را با دو فلگ `MAP_PRIVATE` و `MAP_SHARED` بنویسید و خروجی را مقایسه کنید)

سوال ۷ (۵۰ نمره): برنامه زیر را در نظر بگیرید:

```c
#include <stdio.h>
#include <string.h>

int main(void)
{
  char buff[5];
  int pass = 0;

  printf("\n Enter the password : \n");
  gets(buff);

  if(strcmp(buff, "pass"))
    printf ("\n Wrong Password \n");
  else{
    printf ("\n Correct Password \n");
    pass = 1;
  }
  if(pass == 'a')
    printf ("\n Successful Login as Admin :) \n");
  else if(pass)
    printf("\n Sucessful Login \n");
  else
    printf ("\n Failed to Login \n");

  return 0;
}
```

الف) منطق برنامه فوق به چه صورت است؟ فکر می‌کنید اشکال این برنامه کجاست؟ اگر کد را کمپایل کنیم یک Warninig دریافت می‌کنیم، دلیل آن چیست؟

ب) اکنون که متوجه اشکال برنامه شده‌اید همین کد با دستور زیر compile کنید:

```
gcc -g -fno-stack-protector main.c -o app
```

برنامه app را اجرا نموده و به آن ورودی های مختلف به فرم ...123 بدهید و خروجی را مشاهده کنید.(طول رشته ورودی را مرحله به مرحله افزایش دهید تا جایی که بجای پیام Wrong Password پیام دیگری چاپ شود.) خروجی برنامه چیست؟ اگر در تعریف متغییر pass (خط ۷)بجای int از register int استفاده کنیم باز هم خروجی به همین صورت است؟ در مورد علت وقوع آن‌ها توضیح دهید و خروجی های خود را در گزارش بیاورید.

راهنمایی: برای مشاهده فضای آدرس و مقادیر موجود در متغییر ها می توانید از gdb استفاده کنید. ممکن است بررسی آدرس حافظه‌های buff و local به روشن تر شدن علت وقوع پدیده کمک کند. دستوراتی که ممکن است در gdb نیاز شود:

`list` `break {lineNumber}` `run` `info registers` `info locals` `print &{var}`

برای اطلاعات بیشتر می‌توانید از man page استفاده کنید:

```
man gdb
```

نکته: ممکن است نتیجه اجرای این برنامه در سیستم های متفاوت یکسان نباشد. پاسخ این تکلیف باتوجه به خروجی هایی که از سیستم خودتان ارائه می‌کنید بررسی خواهد شد.

ج) آسیب‌پذیری برنامه را رفع کنید و کد نهایی را پیوست کنید.

شبیه سازی

سوال ۸ (۷۰ نمره) در این تمرین می‌خواهیم کارایی الگوریتم های مختلف page replacement را در دنیای واقعی مقایسه کنیم! برای انجام این مقایسه از شبیه ساز معرفی شده در بخش ۲۲ کتاب درسی استفاده کنم. (ch22:Beyond Physical Memory: Policies) این شبیه‌ساز را می‌توانید از این لینک دریافت کنید.

الف) با استفاده از برنامه `valgrind` آدرس هایی که برنامه ls درخواست می‌کند را در یک فایل بریزید. (ls-trace.txt)

```
valgrind --tool=lackey --trace-mem=yes ls &> ls-trace.txt
```

ب) فرض کنید در فایل ls-trace.txt آدرس های زیر نوشته شده است:

`0xf44ab8aa` `0x044abd8a`

با فرض این که سایز هر صفحه ۴KB است، شماره صفحه مجازی مربوط به هر یک از آدرس های بالا را حساب کنید.

ج) اکنون که نحوه تبدیل آدرس مجازی به شماره صفحه را فراگرفتید می‌توانید از اسکریپت va2vpn.py استفاده کنید که این تبدیل را برای شما انجام می‌دهد و در فایل vpn.txt قرار می دهد.(آرگومان اول نام فایل ورودی است و آرگومان دوم حداکثر تعداد رکوردی است که می‌خواهیم تبدیل کنیم)

```
./va2vpn.py ls-trace.txt 3000
```

د) اکنون باتوجه به فایل های تولید شده به سوالات زیر پاسخ دهید:

۱. تعداد کل درخواست های حافظه از طرف برنامه ls چقدر است؟(می توانید از برنامه wc استفاده کنید)

۲. تعداد صفحات مجزایی که برنامه ls درخواست کرده است چقدر است؟(می توانید از ابزار uniq استفاده کنید)

۳. صفحه گیت هاب مربوط به شبیه ساز را مطالعه کنید تا با نحوه کار کردن با آن آشناشوید سپس با کمک این شبیه ساز و با فرض cache size=3 هر یک از الگوریتم های FIFO, OPT, LRU, UNOPT, RAND, CLOCK را شبیه‌سازی کنید (فایل ورودی= vpn.txt) و مقدار HitRate را برای هرکدام از آن‌ها بدست آورید.(برای شمارش تعداد HIT و MISS می توانید از برنامه grep با سوئیچ c- استفاده کنید)

۴. **(اختیاری ۲۰+ نمره)** نمودار Hit rate بر حسب Cache size را برای هر ۶ الگوریتم رسم کنید و مقایسه کنید.

$CacheSize \in \{1, 2, 3, 4\}$

عکس های مربوط به خروجی شبیه‌ساز را در گزارشتان قرار دهدید و روند انجام کار را به صورت مرحله به مرحله توضیح دهید.

شیوه تحویل

برای این تمرین می بایست یک فلدر به نام studentid_hw2 بسازید (به جای studentid باید شماره دانشجویی خود را قرار دهید) که شامل فایل زیر باشد:

۱. یک فایل pdf: شامل پاسخ به تمام سوالات (ترجیحاً به زبان فارسی) که می بایست با استفاده از LaTeX ایجاد شده باشد.

۲. فایل های C: برای سوال های ۶ و ۷، علاوه بر توضیحی که در فایل pdf فوق در مورد این سوال داده می‌شود باید فایل source code به زبان C نیز در این فلدر قرار داده شود.

سپس فلدر خود را با دستور زیر بایگانی و فشرده سازی کنید.

```
tar zcf studentid_hw2.tgz studentid_hw2
```

و تنها فایل studentid_hw2.tgz را در سامانه یکتا در قسمت مربوط به تکلیف دوم بارگذاری کنید.

موفق باشید