دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

عنوان: تکلیف اول درس سیستم‌های عامل ۱

نام و نام خانوادگی: علیرضا ابره فروش

شماره دانشجویی: ۹۸۱۶۶۰۳

نیم سال تحصیلی: پاییز ۱۴۰۰

مدرّس: دکتر محمّدرضا حیدرپور

دستیاران آموزشی: مجید فرهادی - دانیال مهرآیین - محمّد نعیمی

فهرست مطالب

۱ سوال اول

۱.۱ آ

Table 1:

	Internal Fragmentation	External Fragmentation
1.	In internal fragmentation fixed-sized memory, blocks square measure appointed to process.	In external fragmentation, variable-sized memory blocks square measure appointed to method.
2.	Internal fragmentation happens when the method or process is larger than the memory.	External fragmentation happens when the method or process is removed.
3.	The solution of internal fragmentation is best-fit block.	Solution of external fragmentation is compaction, paging and segmentation.
4.	Internal fragmentation occurs when memory is divided into fixed sized partitions.	External fragmentation occurs when memory is divided into variable size partitions based on the size of processes.
5.	The difference between memory allocated and required space or memory is called Internal fragmentation.	The unused spaces formed between non-contiguous memory fragments are too small to serve a new process, is called External fragmentation .
6.	Internal fragmentation occurs with paging and fixed partitioning.	External fragmentation occurs with segmentation and dynamic partitioning.
7.	It occurs on allocation of a process to a partition greater than the process's requirement. The left over space causes degradation system performance.	It occurs on allocation of a process to a partition greater which is exactly the same memory space as it is required.

۲.۱ ب

سطوح مختلف حافظه مثل کش، هارد درایو، رم و غیره هر کدام سرعت و حجم متفاوتی را ارائه می‌دهند. علی‌رغم اینکه کش سریع‌ترین حافظه است اما نمی‌توانیم که در همه جای سیستم، از کش استفاده کنیم، چون هزینه بالایی دارد. همچنین نمی‌توانیم همه جا از هارد استفاده کنیم چون سرعت کمی دارد و فضای زیادی اشغال می‌کند، درحالی که ارزان است. از سطوح مختلف برای برقراری تعادل میان سرعت و حجم استفاده می‌شود. در cpu که سرعت بالایی دارد ناچاریم از کش‌ها استفاده کنیم و نمی‌توانیم از حافظه‌ی رم کمک بگیریم. نوع حافظه‌ها که بسته به برق اطلاعاتشان پاک می‌شود و احتیاج است که گاهی اطلاعات ماندگار باشد و

گاهی نیازی نیست، یکی از دلایل دیگر آن است. سطوح مختلف حافظه با این هدف شکل گرفته است که داده‌هایی که غالبا توسط برنامه‌ها مورد استفاده قرار می‌گیرند در سطوح بالایی از سلسله مراتب حافظه نگهداری شوند و اغلب درخواست‌ها به حافظه توسط همین سطوح بالاتر مورد پردازش قرار گیرند. در نتیجه حداکثر سرعت را با حداقل هزینه به دست آورده‌ایم.

۳.۱ ج

از آن جایی که اندازه‌ی فضای Stack به صورت پایین‌رونده و فضای Heap به صورت بالارونده حین اجرا ممکن است افزایش یابد، درصورت متوالی بودن conflict رخ می‌دهد.

۴.۱ د

در پروسس‌هایی که نمونه‌های مستقل یک برنامه هستند(درواقع بخش code segment آن‌ها یکسان است)، code segment آن‌ها در مکان‌های یکسان ذخیره شده است و رجیسترهای base و boundِ مربوط به code segment مقادیر یکسان دارند، و stack segment و heap segment دو پروسس مستقل از هم هستند. چون هر دوی این پروسس‌ها به code segment دسترسی دارند باید اجازه نوشتن در این قسمت از آن‌ها سلب شود تا در عملکرد دیگری اختلال ایجاد نکند و این پشتیبانیِ سخت‌افزاری را می‌طلبد. در نتیجه اطلاعات مربوط به protection که اجازه‌های خواندن، نوشتن و اجرا هستند را به هر segment اضافه می‌کنیم.

۵.۱ ه

بزرگی اندازه‌ی Paging موجب کوچک شدن page tableها می‌شود. درنتیجه احتمال وقوع page fault را کاهش می‌دهد. همچنین میزان سربار کمتر است. اما internal fragmentation در این حالت بیشتر است. همچنین locality of reference نیز کاهش می‌یابد. کوچکی اندازه‌ی Paging، internal fragmentation را کاهش می‌دهد و همچنین locality of reference نیز افزایش می‌یابد. اما در این حالت page tableها بزرگتر هستند و درنتیجه احتمال وقوع page fault را بیشتر است. از طرفی دیگر میزان سربار بیشتر است.

۶.۱ و

سربار ناشی از انجام این عملیات توسط سخت‌افزار زیاد است، چون اولا دسترسی به بسیاری از جزئیات(برای مثال swap space) به شکل سخت‌افزاری به شدت پیچیده می‌شود. همچنین دیسک کند است و عملیاتِ انجام گرفته توسط آن نسبت به حالت نرم‌افزاری بسیار زمان‌بر است. پس رسیدگی به آن به صورت نرم‌افزاری به‌صرفه‌تر است.

۷.۱ ز

رجیسترِ CR3 برای هر پروسس page table baseِ مربوط به آن پروسس را مشخص می‌کند.

۸.۱ ح

با توجه به منبع شماره‌ی ۱ در قسمت منابع به ترتیب داریم:

- درصورتی که آدرس ترجمه شده در TLB وجود نداشته باشد TLB Miss رخ می‌دهد. پس به PTE مربوطه مراجعه می‌شود و درصورت set بودن بیت valid(present)، page fault رخ نمی‌دهد.

- درصورتی که به PTEای که بیت validِ آن صفر است اشاره کند، دسترسی غیرمجاز به حافظه تلقی می‌شود. اما اگر PTE وجود نداشته باشد نشانگر این است که صفحه در physical memory address space قرار ندارد.
- درصورتی که TLB Hit رخ دهد و بیتِ validِ PTE نظیر آن یک باشد page fault رخ نخواهد داد.
- فرض که TLB Hit رخ داده است، اما حین دسترسی به یک trap برمی‌خوریم که احتمالا به دلیل این است که تلاش می‌کنیم به قسمتی که Read-only است دسترسی پیدا کنیم.

۲ سوال دوم

Table 2: LRU

Access	Hit	State(after)
2	no	2
3	no	2, 3
1	no	2, 3, 1
5	no	2, 3, 1, 5
6	no	3, 1, 5, 6
2	no	1, 5, 6, 2
1	yes	5, 6, 2, 1
5	yes	6, 2, 1, 5
3	no	2, 1, 5, 3
2	yes	1, 5, 3, 2
6	no	5, 3, 2, 6
5	yes	3, 2, 6, 5
4	no	2, 6, 5, 4
3	no	6, 5, 4, 3
2	no	5, 4, 3, 2

Table 3: LFU

Access	Hit	State(after)
2	no	2
3	no	2, 3
1	no	2, 3, 1
5	no	2, 3, 1, 5
6	no	6, 3, 1, 5
2	no	2, 6, 1, 5
1	yes	2, 6, 1, 5
5	yes	2, 6, 1, 5
3	no	3, 2, 1, 5
2	yes	3, 2, 1, 5
6	no	6, 2, 1, 5
5	yes	6, 2, 1, 5
4	no	4, 2, 1, 5
3	no	3, 2, 1, 5
2	yes	3, 2, 1, 5

Table 4: Optimal

Access	Hit	State(after)
2	no	2
3	no	2, 3
1	no	1, 2, 3
5	no	3, 5, 1, 2
6	no	6, 5, 1, 2
2	yes	6, 2, 5, 1
1	yes	1, 6, 2, 5
5	yes	1, 5, 6, 2
3	no	3, 5, 6, 2
2	yes	3, 5, 6, 2
6	yes	6, 2, 3, 5
5	yes	6, 5, 2, 3
4	no	5, 4, 2, 3
3	yes	5, 4, 2, 3
2	yes	5, 4, 2, 3

۳ سوال سوم

ابتدا بررسی می‌کنیم که TLB Miss رخ داده است یا TLB Hit. این کار در ۱ نانو ثانیه انجام می‌شود. سپس دو حالت زیر را در نظر می‌گیریم:

- TLB Miss رخ دهد. در این حالت داریم: $P_{TLBMiss}(2 * T_{DRAM} + P_{PageFault} * T_{Disk}) = 4ns$
- TLB Miss رخ ندهد. در این حالت داریم: $(1 - P_{TLBMiss}) * (T_{Cache} + P_{CacheMiss} * P_{PageFault} * T_{Disk}) = 2.97ns$

در نهایت میانگین زمان دسترسی به صفحات حافظه برابر است با: $1ns + 4ns + 2.97ns = 7.97ns$

۴ سوال چهارم

۱.۴ الف

گام اول: تجزیه‌ی آدرس مجازی به Seg، VPN، و Offset:

۲ بیت پرارزش مربوط به سگمنت است، چون صفحه‌ها ۳۲ بایتی هستند پس ۵ بیت برای نشان دادن آن‌ها نیاز داریم که از راست برای آفست جدا می‌کنیم و ۵ بیت باقی‌مانده برای VPN است.

0x45d = 01 00010 11101

SN = 01

VPN = 00010 = 2

Offset = 11101 = 29

Address of PTE = Base[SN] + VPN*sizeof(PTE) = 64 + 2*1 = 66

بایت ۶۶ در صفحه‌ی با PFN = 2 قرار دارد و بایت سوم آن است.

Page with PFN = 2: c3

c3 = 1 1000011 = 67

valid bit = 1

PFN = 67

آفست ۲۹ در صفحه‌ی ۶۷ که آدرس ۴۰ است. پس برای ترجمه آدرس مجازی به آدرس فیزیکی، به صفحات ۲ و ۶۷ رجوع می‌شود. بله چون بیتِ valid یک است و بایت مورد نظر c3 است که در صفحه‌ی ۲ قرار دارد.

۲.۴ ب

گام اول: تجزیه‌ی آدرس مجازی به Seg، VPN، و Offset:

۲ بیت پرارزش مربوط به سگمنت است، چون صفحه‌ها ۳۲ بایتی هستند پس ۵ بیت برای نشان دادن آن‌ها نیاز داریم که از راست برای آفست جدا می‌کنیم و ۵ بیت باقی‌مانده برای VPN است.

0xc85 = 11 00100 00101

SN = 11

VPN = 00100 = 4

Offset = 00101 = 5

Address of PTE = Base[SN] + VPN*sizeof(PTE) = 640 + 4*1 = 644

بایت ۶۴۴ در صفحه‌ی با PFN = 20 قرار دارد و بایت پنجم آن است.

Page with PFN = 20: 20

20 = 0 0100000

valid bit = 0

خیر چون بیتِ valid صفر است، این آدرس مجازی به این پروسس تخصیص داده نشده است و تنها به صفحه‌ی ۲۰ رجوع می‌شود.

منابع

[1] https://gateoverflow.in/150841/Tlb-and-page-fault