حالت محفاظت شده

حالت حفاظت شده چیست؟

پردارنده های ۸۰۸۶ که در کامپیوترهای آی بی ام استفاده میشده است قابلیت انعطاف بسیار پایینی داشتند مخصوصاً اینکه راه راحت و تعریف شده ای برای دسترسی به بیش از ۱ مگابایت حافظه در آن وجود نداشت.برای حل این مشکل شرکت اینتل ۸۰۲۸۶ را طراحی کرد و بابت اینکه بتواند از سلف خود یعنی ۸۰۸۶ پشتیبانی کند دو حالت واقعی(real mode) و حالت حفاظت شده(protected mode) را در آن پشبینی کرد(16-bit protect mode).
در حالت حفاطت شده برنامه ها حق استفاده بیشتر از ۱ مگابایت حافظه را دارند و دیگر مزیت استفاده از این حالت در این میباشد که برنامه های دیگر حق استفاده از حافظه ای که به شما تخصیص داده شده را ندارند.حالت دیگر محافظت شده که ۳۲ بیت هم میباشد در ۳۸۶ و بالاتر وجود دارد درواقع تفاوت اصلی فقط در آدرسی دهی آن میباشد و مفاهیم یکی میباشد.

تفاوتهای حالت محافظت شده و حالت واقعی

	حالت واقعی	حالت محافظت شده ۱۶ بیت	حالت محافظت شده ۳۲ بیت
آدرس پایه قطعه(segment)	20 bit (1 mb) = 16 * segment register	24 bit(16 mb), From descriptor	32 bit(4 gb), From descriptor
اندازه قطعه	16 bit,64Kb fixed	16 bit,1-64Kb	20 bit 1Mb,4Kb – 4Gbytes
حفاظت از قطعه	نه	بله	بله
ثبات قطعه	آدرس پایه قطعه/۱۶	Selector	Selector

من فکر میکردم در حالت حفاظت شده از قطعه استفاده نمیشود

Segmentها هنوز استفاده میشوند اما در حالت حفاظت شده ۳۲ بیت شما میتوانید اندازه یک قطعه را به 4 گیگابایت افزایش دهید در واقع این عدد آخرین حد استفاده از رم در کامپیوترهای ۳۲ بیت میباشد.با این روش segment را حذف کرده ایم(بهرحال خاصیت حالت حفاظت شده از بین نرفته است). این روش باعث محبوبیت پردازنده های ۳۲ بیت شد.

توصیفگر (Descriptor) چیست؟

در حالت واقعی شما نیاز به اطلاعات زیادی در رابطه با قطعه لازم ندارید.در واقع تمام آنها به اندازه ۶۴ کیلوبایت میباشند و شما هر کار که میخواهید میتوانید با آنها انجام دهید مانند: ذخیره داده ، استفاده بعنوان پشته و یا حتی قرار دادن کد اجرایی خود در آن.
و آدرس پایه قطعه هم بدین صورت میباشد: ۱۶ ضربدر مقدار یکی از ثباتهای قطعه.
درحالت حفاظت شده علاوه بر اینکه باید آدرس پایه قطعه را بدانیم همچنین باید اندازه قطعه و دیگر نشانه ها(flag) را که به ما نشان میدهد آن قطعه برای چکاری هست را بدانیم.این اطلاعات در یک ساختار داده ۸ بایتی ذخیره میشود که به آن descriptor میگویند.

جدول ۲:code/data segment descriptor

Lowest byte	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Highest byte**
Limit 7:0	**Limit 15:8**	**Base 7:0**	**Base 15:8**	**Base 23:16**	**Access**	**Flags,limit 19:16**	**Base 31:24**

این توصیفگر ۳۲ بیت میباشد.

توصیفگر ۱۶ بیت(۸۰۲۸۶) باید دو بایت در بالا داشته باشد که صفر شده باشند.
(flags,limit 19:16,base 31:24)
بایت access مشخص کننده استفاده قطعه میباشد
(data segment,stack segment,code segment,…)

جدول ۳:access byte of code/data segment descriptor

Highest byte	Bit 6,5	Bit 4	Bit 3	Bit 2	Bit 1	Lowest byte
present	Privilege	1	Executable	Expansion irection/confoming	Writeable/readable	Accessed

Present بیت باید یه ۱ تبدیل شود بایت دستیابی قطعه
Privilege صفر بالاترین سطح از privilege میباشد(0 ring) و پایین سطح آن نیز ring) (3 میباشد.
Executable بیت . اگر یک باشد قطعه،قطعه کد میباشد و غیر از آن قطعه،قطعه پشته و یا داده میباشد.
Expansion direction (قطعه پشته/داده). اگر یک باشد ، قطعه بسمت پایین رشد میکند و افست بین قطعه باید بزرگتر از حد خود باشد.
Conforming(قطعه کد).بستگی به privilege دارد.
Writeable(پشته/قطعه داده).اگر یک باشد قطعه میتواند نوشته شود.
Readable(قطعه کد) . اگر یک باشد قطعه میتواند از جایی خوانده شود(قطعه کد قابل نوشتن نیستwriteable)
Accessed.این بیت هرگاه که قطعه خوانده یا نوشته میشود مقدار دهی میشود.

نشانه های (flag) ۴ بیتی فقط برای قطعات ۳۲ بیتی مقدارشان صفر نیست.

جدول ۴: نشانه های ۴ بیتی یا نیبل(nibble)

Highest bit	Bit 6	Bit 5	Bit 4
granularity	Default size	0	0

بیت گرانولیته(granularity) اگر اندازهٔ قطعه در واحدهای ۴ کیلویی باشد بصورت G=1 درمی آید و اگر اندازه قطعه در واحد بایت باشد بصورت G=0 میباشد.
برای قطعه پشته مقدار پیش فرض بیت بصورت B(big) شناخته میشود،و هر مقدار ۳۲ بیتی و ۱۶ بیتی را که در پشته ذخیره شده است و یا از پشته در حال خواندن هست را کنترل میکند.
برای قطعه کد، بیت D هر دستورالعملی را که میخواهد عملیاتی بر روی ۱۶ بیتی (D=0) یا ۳۲ بیتی (D=1) انجام دهد را نشان میدهد. برای فهم بیشتر به این مثال توجه کنید:
وقتی بیت D به یک ست میشود یعنی اینکه دستورالعمل ۳۲ بیتی هست و به قطعه کد میگوید که بصورت ۳۲ بیتی عمل کند و اسمبلر از دستور کمکی USE32 برای فهم این موضوع استفاده میکند به رشته زیر توجه کنید:

B8 90 90 90 90

پردازنده با این دستور العملهای بصورت ۳۲ بیت رفتار خواهد کرد وبه این صورت disassemble خواهد شد :

Mov eax,90909090h

در دستورالعملهای ۱۶ بیتی از دستور کمکی USE16 در کد قطعه استفاده میشود همان رشته ای که در بالا به آنها اشاره شد بدین صورت خواهد بود:

Mov ax,9090h
Nop
Nop

دو بایت دستورالعملهای مخصوص ماشین(opcode) به نامهای Operand Size Prefix و Address Length Prefix حالت بیتD را برای دستورالعملهای مقصد و مبدا معکوس میکنند.این پیشوندها فقط روی دستورالعملهایی تاثیر میگذارند که بدون واسطه آنها را دنبال کنند.
بیت ۴ از بیتAccess برای قطعه کد یا داده/پشته به مقدار ۱ تنظیم شده است.اگر این مقدار ۰ باشد شما یک قطعه سیستم (system segment) دارید.این مقدار در چند حالت وجود دارد:

- Task State Segment(Tss) این قطعه برای راحت تر کردن کار در محیط multitasking میباشد.پردازنده های ۸۰۳۸۶ و بالاتر ۴ نوع دیگر از این نوع قطعه را دارا میباشند.
- Local Descriptor Table(LDT) . وظایف در حال انجام در پردازنده میتوانند توصیفگرهای اختصاصی خود را در اینجا ذخیره کنند بجای GDT
- Gates تغییر وضیعتهای پردازنده را کنترل کرده که از یک سطح privilege به سطح دیگر میرود.توصیفگر Gates ساختار متفاوتی از بقیه توصیفگرها دارد.

جدول ۵:Gates Descriptor

Lowest byte	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Highest byt
Offset 7:0	Offset 15:8	Selector 7:0	Selector 15:8	Word count 4:0	Access	Offset 23:16	Offset 31:24

توجه کنید به که رکورد Gates.selector بصورت غیرمستقیم کار میکند و احتیاج به یک کد مستقل یا توصیفگر TSS برای کار کردن دارد.

جدول ۶:access byte of system segment descriptor

Highest bit	Bit 6,5	Bit 4	Bits 3,2,1,0
present	Privilege	0	Type

جدول ۷:System Segment types

Type	Segment function	Type	Segment function
0	(invalid)	8	(invalid)
1	Available 286 Tss	9	Available 386 TSS
2	LDT	10	(undefined, reserved)
3	Busy 286 TSS	11	Busy 386 TSS
4	286 call Gate	12	386 call Gate
5	Task Gate	13	(undefined, reserved)
6	286 interrupt Gate	14	386 Interrupt Gate
7	286 Trap Gate	15	386 Trap Gate

چه اطلاعاتی سختی! شما فقط باید اینرا به خاطر بسپارید که TSSها وLDTها و Gateها سه مدل اصلی قطعه سیستمی میباشند.

توصیفگرهای کجا میباشند؟

آنها در جدول حافظه ذخیره میشوند در Global Descriptor Table(GDT) ، Interrupt Descriptor Table(IDT) و یا یکی از جداول توصیفگر داخلی (Local Descriptor Table)
پردازنده حاوی سه ثبات میباشد :

- GDTR که اشاره به GDT دارد
- IDTR که اشاره به IDT دارد(البته این درصورتی است که از وقفها استفاده شده باشد)
- LDTR که اشاره به LDT دارد(اگر LDT استفاده شده باشد)

هر کدام از این جداول میتونند تا ۸۱۹۲ توصیفگر را در خود جای دهند.

انتخابگر (Selector) چیست؟

در حالت حفاضت شده، ثباتهای قطعه حاوی selectorها هستند که در یکی از جداول توصیفگر اندیس شده اند.فقط ۱۳ بیت از selectorها برای این اندیس استفاده میشود.
بیت پایینی بعدی بین GDT و LDT انتخاب میشود.دو بیتی که در پایین ترین جایگاه در selector قرار دارند بابت تعیین مقدار privilege میباشند.

چگونه وارد حالت محافظت شده شوم؟

وارد شدن به حالت محافظت شده در واقع ساده تر از توضیحات آن میباشد!!!

- یک Global Descriptor Table(GDT) معتبر بسازید.
- (انتخابی) یک Interrupt Descriptor Table(IDT) معتبر بسازید.
- وقفه ها را غیر فعال کنید.
- ثبات GDTR را به GDT خود اشاره دهید.
- (انتخابی) ثبات IDTR را به IDT که ساخته اید اشاره دهید.
- بیت PE را در ثبات MSW قرار دهید.
- یک پرش دور (منظور از دور یعنی اینکه در قطعه ای پرش انجام میشود نباشد)(دو ثبات CS و IP/EIP رو بار کنید) به حالت حفاظت شده انجام بدهید(ثبات CS را با انتخابگر قطعه کد بار کنید).
- ثبات های DS و SS را با انتخابگر قطعه داده/پشته بار کنید.
- پشته حالت محافظت شده را راه اندازی کنید.
- (انتخابی) وقفه ها را فعال کنید.

چگونه به حالت واقعی برگردم؟

در پردازنده های ۳۸۶ و بالاتر:

- وقفه ها را غیر فعال کنید.
- یک پرش دور به یک قطعه کد ۱۶ بیتی انجام بدهید
- ثبات SS را با انتخابگر به قطعه داده/پشته ۱۶ بیتی بار کنید.
- بیت PE را پاک کنید.
- یک پرش دور به یک آدرس در حالت واقعی انجام بدهید.
- ثباتهای DS,ES,FS,GS و SS را در حالت واقعی مقدار دهی کنید.
- (انتخابی) IDTR را در حالت واقعی مقدار دهی کنید(پایه ۰ و حد آن 0xFFFF)
- وقفه ها را فعال کنید.

قبل از اینکه به حالت واقعی برگردید، ثباتهای CSوSS باید حاوی انتخابگرهایی که به توصیفگرهای حالت واقعی اختصاص دارند اشاره میکنند باشند.

بر روی ۸۰۲۸۶ شما نمیتوانید به این راحتی ها بیت PE را پاک کرده و به حالت واقعی برگردید!تنها راه ممکن ریست کردن پردازنده میباشد.
برای اینکار میتوانید سیگنال ریست کردن را به کیبورد بفرستید و یا خطای سه گانه ای در پردازنده بوجود آورید(برای اطلاعات بیشتر به وب سایت روبرت کولین مراجعه کنید www.x86.org)

وارد چه دامی شده اید؟

- شما باید واقعاً حواستان را جمع کرده و جزیئات را بخوبی بررسی کنید.یک بیت اشتباه میتواند همه چیز را خراب کند.خطاهای حالت محافظت شده معمولاً باعث خطاهای سه گانه در پردازنده میشود و باعث میشود پردازنده خودش را ریست کند. آمادگی برای این مشکل را داشته باشید نه یکبار بلکه چندین دفعه!
- خیلی از روالهای کتابخانه ای مطمئناً کار نخواهند کرد مانند روال printf() بخاطر اینکه احتمالاً از سرویسهای DOS و یا BIOS استفاده میکند.مگراینکه توسعه دهنده داس داشته باشید(DOS extender).
- قبل از پاک کردن بیت PE ثباتهای قطعه باید به توصیفگرهایی که به حالت واقعی دارند اشاره کند.این دقیقاً محدودیت 0xFFFF را نشان میدهد.

درحقیقت برای ثباتهای DS,ES,FS و GS حد قطعه باید 0xFFFF و یا بیشتر باشد. اگر شما برروی قطعه این محدودیت را اعمال کنید و آن page-granular کنید شما میتوانید به ۴ گیگابایت از حافظه دسترسی پیدا کنید در حالت واقعی. این حالت ، حالتِ unreal mode لقب گرفته است.اگر چه محدودیت ها و page-granularity در ثباتهای CS و یا SS باعث بوجود آمدن مشکل بزرگی در حالت واقعی میشود.

- شما نمیتوانید از دستورالعمل LMSW که در ۸۰۲۰۸۶ موجود میباشد برای پاک کردن بیت PE استفاده کنید. باید از دستور MOV CR0,nnn استفاده کنید.
- بعد از ورود به حالت حفاظت شده تمام ثباتها را باید با انتخابگرهای معتبر بار کنید. یک روال در حالت محافظت شده ES را در پشته نگه میدارد و آن را با یک انتخاگر معتبر پر میکند و از آن استفاده میکند و وقتی سعی میکند آنرا میخواهد از پشته دربیاورد انتخابگر غیرمعتبر در ثبات ES مینشیند و باعث ازکار افتادن سیستم میشود.
- IDTR باید به مقادیری که به حالت واقعی اختصاص دارند برگردانده شود قبل از اینکه بخواهیم وقفه ها را دوباره فعال کنیم.
- تمام دستورالعمل ها در حالت واقعی مجاز نیستند اگر شما بخواهید که از قطعه وضعیت(state segment) برای چندوظیفه ای استفاده کنید توجه داشته باشید که اجرای دستورالعملهای LTR در حالت واقعی باعث اجرای وفقه های اشتباه خواهد شد.

GDT و همچنین LDT باید در حافظه مستقر شوند بدین دلیل که پردازنده بیت access که مربوط به توصیفگر میباشد را تغییر میدهد.

- کدهای خامی که در اینجا آنها را توضیح دادیم امکان دارد در حالت مجازی ۸۰۸۶ پردازنده شما را از کار بیندازد.این چهارمین حالت از عملکرد پردازنده های ۸۰۳۸۶ میباشد که آدرس دهی در آن شبیه آدرس دهی در حالت واقعی مباشد ولی یکسری از مکانیزمهای حالت حفاظت شده وجود دارد.

اگر شما میخواهید شروع کنید این نکات ریز را بکار گیرید:

- نگران برگشت به حالت واقعی نباشید از دکمه ریست استفاده کنید ☺
- وقفه ها را غیرفعال نگه دارید
- از LDT استفاده نکنید
- فقط ۴ توصیفگر در GDT قرار بدهید:null,code,stack/data and text video
- مقدار پایه قطعه را در حالت واقعی مقدار دهی کنید مثال : 16*real-mode segment register value
- تمام قطعات باید به حدِ گفته شده یعنی 0xFFFF تنظیم شوند(برای حالت حفاظت شده ۱۶ بیتی)
- تمامی مقادیر privilege باید به ۰ تنظیم شود(Ring 0)

به کد زیر توجه کنید:

```
Void unhand(void)
{
	Static const char msg[]="U n h a n d l e d I n t e r r u p t ";

	Disable();
	Movedata(SYS_DATA_SEL, (unsigned)Msg,
			LINEAR_SEL,0xB8000,
			Sizeof(Msg));
	While(1); }
```

فضاهای خالی که در متن بالا میبینید مانند کد خاصیت کاراکترها در پردازنده گرافیکی عمل میکند با قراردان یک وقفه Gate در توصیفگر خاصی که مربوط به IDT و با یک انتخابگر به قطعه کد شما در trap gate و آدرس این روال در افست مربوطه این عمل انجام میشود.

توضیحات

Dos extender: توسعه دهنده داس:
برنامه ای که برای گسترش حافظه معمولی ۶۴۰ کیلوبایتی کامپیوترها طراحی میشود تا خود داس و برنامه های کاربردی مبتنی بر داس از آن استفاده کنند. این برنامه این کار را از طریق کاربرد قسمتی از حافظه رزرو شده انجام میدهد.

Privilege instruction:
دستورالعملی(معمولاً به زبان ماشین)که تنها سیستم عامل میتواند آن را اجرا نماید.دلیل وجود این گونه دستورالعملها آن است که سیستم عامل نیاز به دستورالعمل هایی دارد که برنامه های کاربردی دیگر نباید بتوانند اجرا نمایند.بنابراین تمام روتینهای سیستم عامل حق اجرای این گونه دستورالعملها را دارند

Privilege mode:
یک مد اجرایی در مد محافظت شده ریزپردازنده های ۸۰۲۸۶ و بالاتر اینتل که نرم افزارها میتوانند عملیات محدود شده ای را انجام دهند که به مدیریت قطعات حیاتی چون حافظه و پورتهای ورودی/خروجی اختصاص دارند.برنامه های کاربردی را نمیتوان در این مد اجرا کرد . نرم افزارهای راه اندازی وسایل متصل به سیستم و همین طور هسته اصلی سیستم عامل os/2 را میتوان در این مد اجرا کرد.

Translated By NETSPC
os@persiasecure.com
netspc@gmail.com
http://groups.google.com/Persian_OS