اصول برنامه‌نویسی اسمبلی

CPU تراشه enCorRe دستور پشتیبانی می‌کند. همه برنامه‌ها باید از این ۳۷ دستور استفاده کنند. سیپرس یک مترجم مجانی ارائه می‌دهد که کدهای اسمبلی را که شما می‌نویسید به فایل‌های موضوع، که به منظور برنامه‌ریزی در EPROM تراشه تهیه می‌شوند، تبدیل می‌کند. اگر ترجیح دهید که در C برنامه‌نویسی کنید، سیپریس یک مفسر C نیز پیشنهاد می‌کند.

اگر با برنامه‌نویسی اسمبلی میکروکنترلر آشنایی داشته باشید، برنامه‌نویسی برای enCoRo نیز مشابه همان است. اما اگر با برنامه‌نویسی در بیسیک و C آشنا هستید، باید بدانید که در برنامه‌نویسی کدهای اسمبلی بسیاری از عملگرهای زبانهای سطح بالا موجود نیست در اینجا دیگر حلقه‌های While یا for یا انواع مختلف متغیرها وجود ندارد. اما برای تراشه‌ای مانند enCoRo که به منظور کارهای نمایشی و کنترلی غیر پیچیده طراحی شده است، استفاده از کدهای اسمبلی عملی است. برای برنامه‌های کوتاه، که به سرعت اجرا می‌شوند احتیاجی به خرید مفسر نیست.

اصول برنامه‌نویسی اسمبلی

برنامه‌نویسی اسمبلی شامل یک مجموعه از دستورات است که هر کدام مربوط به کدهای ماشینی هستند که تراشه از آنها پشتیبانی می‌کند. مثلاً دستور iord، که محل io را می‌خواند به کد ۲۹h مربوط است. به جای به خاطر آوردن ۲۹ h، شما می‌توانید iord را بنویسید، و مترجم معادل سازی را برای شما انجام خواهد داد. دستور iord همچنین احتیاج به یک عملوند دارد که محل خواندن را مشخص کند. به عنوان مثال iord 01h پورتی با آدرس ۱۰ h را می‌خواند.

زبان برنامه‌نویسی اسمبلی همچنین می‌تواند شامل دایرکتیو[1] و توضیحات باشد. دایرکتیوها دستوراتی هستند که به جای اینکه مربوط به CPU باشند، مربوط به مترجم می‌باشند. دایرکتیوها شما را قادر می‌سازند که محلی از حافظه را مشخص کنید، متغیرهایی تعریف نمایید. در کل، نقشی که مترجم در کنار اجرای دستورات مشخص شده باید ایفا کند را نشان می‌دهند. یک نقطه ویرگول (:)یا ممیز دوبل (//) یک عبارت توصیفی را مشخص می‌کنند که مترجم از آنها چشم‌پوشی می‌کند.

[1] - Directive

مترجمی که توسط سیپرس ارائه می‌شود، cyasm.exe قابل اجرا در پنجره داس[1] می‌باشد. سیپرس مرجع‌ها و راهنمای استفاده برای کاربرانی را تهیه کرده است که چگونگی استفاده از مترجم را شرح می‌دهد.

مترجم از دو مجموعه دستور مشابه برای CPUهای سری A و سریB پشتیبانی می‌کند. تراشه‌های enCoRo از سری B هستند. تراشه‌های قدیمی‌تر سیپرس، مانند ۶۳۰۰۱، از سری A بودند و از همهٔ دستورات بجز بعضی از آنها پشتیبانی می‌کنند.

کدهای مترجم

راهنمای کاربران دارای توضیحات کاملی در مورد کد اسمبلی و دایرکتیوهاست و در اینجا برخی از جزئیات آن تکرار می‌شود. جدول ۱-۸ خلاصه‌ای از کدها می‌باشد و جدول ۲-۸ خلاصه‌ای از دایرکتیوها را نشان می‌دهد. کدهای ماشین تراشه به ۳۷ دستور ترجمه شده است.

[1]- Dos

جدول ۱-۸: متجرم Cyasm از ۳۷ دستور اسمبلی برای enCoRo پشتیبانی می‌کند

نوع دستور	دستور	توضیح
تابعهای منطقی و ریاضی	ADD	اضافه کردن بدون نقلی
	ADC	اضافه کردن همراه با نقلی
	AND	AND کردن بیتی
	ASL	انتقال به چپ منطقی
	ASR	انتقال به راست منطقی
	CMP	مقایسه
	CPL	متمم کردن آکومولاتور
	DEC	کاهش
	INC	افزایش
	OR	OR کردن بیتی
	RLC	چرخش به چپ همراه با نقلی
	RRC	چرخش به راست همراه با نقلی
	SUB	تفریق بدون نقلی
	SBB	تفریق همراه با نقلی

	XOR	OXR بیتی
پرشهای برنامه و کنترلی	GALL	فراخوانی تابع
	HALT	اجرای ایست
	RETI	بازگشت از وقفه
	JACC	پرش آکومولاتور
	JC	پرش در صورتی که نقلی یک باشد
	JMP	پرش
	JNC	پرش در صورتی که نقلی صفر باشد
	JNZ	پرش اگر صفر نباشد
	JZ	پرش اگر صفر باشد
	RET	بازگشت
	XPAGE	صفحه حافظه

ادامه جدول ۱-۸: مترجم Cyasm از ۳۷ دستور اسمبلی برای enCoRo پشتیبانی می‌کند

نوع دستور	دستور	توضیح
انتقال داده	INDEX	خواندن جدول
	IORD	خواندن از I/O
	IOWR	نوشتن بر I/O
	IOWX	مشخص کردن نوشتن I/O
	MOV	انتقال
	POP	انتقال داده از پشته به آکومولاتور
	PUSH	انتقال داده از آکومولاتور به پشته
	SWAP	SWAP
بقیه	DI	غیر فعال کردن وقفه‌ها
	EI	فعال کردن وقفه‌ها
	NOP	بدون عمل

این دستورات، تابعهای اصلی ریاضی و منطقی، پرسشهای برنامه و کپی داده‌ها از رجیسترها، پورت‌ها و RAM را انجام می‌دهند، دو بیت پرچ نقلی[1] و صفر اطلاعات

[1] - carry

بیشتری را ارائه می‌دهند. مانند اینکه نتیجه دستور add دارای سرریز[1] بوده است یـا خیـر یا اینکه نتیجه صفر شده است یا نه.

تراشه از سه حالت آدرس دهی پشتیبانی می‌کند که چگونگی استفاده از عملوند را بـرای دستور مشخص می‌کند. همه دستورات از هر سه حالت آدرس دهی پشتیبانی نمی‌کنند.

در آدرس دهی سریع، دستورات از مقدار عملوند مستقیماً استفاده می‌کنند. این دسـتور از آدرس دهی سریع برای جمع کردن ٦٠ h با مقدار آکومولاتور استفاده می‌کند.

Add A/ 60h

در آدرس دهی مستقیم، دستور با عملوند شبیه به آدرس رفتار می‌کند و از مقداری که در آن آدرس ذخیره شده استفاده می‌نماید. این دستور از آدرس دهـی مسـتقیم بـرای جمـع کردن مقداری که در آدرس ٦٠h از RAM نوشته شده با محتویات آکومولاتـور اسـتفاده می‌کند.

Add A/ [60h]

در آدرس دهی شاخصی، دستور از داده‌ای استفاده می‌کند که در آدرس حاصل از افزودنه یک مقدار به رجیستر X قرار گرفته است. آدرس دهی شاخصـی بـرای کپـی کـردن یـک بلاک از داده مفید می‌باشد. رجیستر X آدرس آغاز بلاک را در خود ذخیره می‌نماید. کـد مقداری را به محتوای رجیستر X اضافه می‌کند تا آدرس بایتی که می‌خواهد کپی شود را به دست آورد. با افزایش این مقدار در هرکپی، کد می‌تواند یک بلاک داده را کپی کند.

[1]-Overflow

استفاده از مترجم

مترجم یک برنامه تحت داس می‌باشد. این دستور:

cyasm test.asm

فایل test.asm را اسمبل می‌کند.

مترجم سه فایل ایجاد می‌کند:

test.asm که کدهای اسمبل شده‌ای برای استفاده در کیت ارتقا هستند، شما می‌توانید از این فایل برای بارگذاری کدها از کامپیوتر بهRAM برد ارتقا استفاده کنید.

در اینجا بخشی از فایل rom، هنگامی که در وایرشگر متنی باز می‌شود نشان داده شده است:

80 99 80 10 80 15 81 24
80 8C 80 99 80 85 80 10
2D 1A 20 1E 20 2D 2A 21
1A 37 16 00 A0 20 27 37

جدول ۲-۸: مترجم cyasm از ۱۳ دایرکتیو پشتیبانی می‌کند

دایرکتیو	توضیح
CPU	مرجع خصوصیات محصول
DB	تعریف بایت
DS	تعریف یک رشته اسکی

DSU	تعریف یک رشته، unicode
DW	تعریف یک کلمه (۲بایت)
DWL	تعریف یک کلمه با بعضی خصوصیات
EQU	برابری نشانه‌ای با یک مقدار متغیر
FILLROM	تعریف مقدار برای حافظه برنامهٔ بی‌استفاده
INCLUDE	اضافه کردن یک فایل
MACRO	تعریف ماکرو
ORG	مبدأ برنامه
XPAGEON	فعال کردن XPAGE
XPAGEOFF	غیر فعال کردن XPAGE

این فایل حاوی خطوطی است که از هشت بایت هگزاسکی[1] با استفاده از فاصله‌ای بین هر کدام تشکیل شده است.

در قالب‌بندی هگزاسکی، هر بایت با دو کد اسکی نمایش داده می‌شود که هر کد یک کاراکتر هگزادسیمال را نشان می‌دهد. مثلاً، بایت ۸۰ h با کدهای اسکی ۳۸ h برای ۸ و ۳۰h برای صفر نمایش داده می‌شود. استفاده از قالب‌بندی هگزاسکی شما را قادر می‌سازد که به راحتی در ویرایشگر متنی مقادیر بایت را ببینید (مثلاً ۸۰). وقتی که کد در

[1] - ASCII hex

RAM برد ارتقا ذخیره شود. RAM شامل بایت‌های باینری می‌شود که توسط بایت‌های هگزاسکی نشان داده شده است. مثلاً، h ۸۰ به ۱۰۰۰۰۰۰۰ در باینری ترجمه می‌گردد.

test.hex کدهای اسمبلی در قالب‌بندی هگزاینتل می‌باشد. بسیاری از برنامه‌ریزان EPROMها از جمله Hi-Lo شرکت سیپرس، از این قالب‌بندی پشتیبانی می‌کنند. کیت ارتقاء می‌تواند به جای قالب‌بندی rom از این قالب‌بندی استفاده کند. قالب‌بندی هگزاینتل از کاراکترهای هگزاسکی و اطلاعات آدرس دهی استفاده می‌کند که در اینجا داده‌هایی که در یک خط فایل hex.* موجود است را می‌بینید.

2000000080998010801581248080C8099808580102D1A201E202D2A211
A371600A0202737A1

test.hex یک فایل لیستی است که توسط مترجم ایجاد می‌گردد. این فایل هر خط کد اسمبلی و توضیحات را نشان می‌دهد. در ادامهٔ آنها، کدهای برنامه معادل و آدرسی که باید در آنجا ذخیره شوند نمایش داده می‌شود. وقتی که از برنامه‌های نمایشی استفاده می‌کنیم. این فایل لیستی، مفید است. مثلاً اگر می‌خواهید که در یک نقطه اجرای برنامه را ثابت کنید، می‌توانید از این فایل لیستی برای پیدا کردن آدرس وابسته به آن خط استفاده کنید.

ستون سمت چپ، آدرس در حافظه برنامه را مشخص می‌کند. این آدرس وقتی که خط، فقط شامل توضیح یا برچسب است تغییر نمی‌کند. دو ستون بعدی بایت‌های ذخیره شده

در هر آدرس می‌باشد. مثلاً، در محل ۰۳CD، مقدار Ah ۲ کدی برای iowr و h ۱٤ نشان دهندهٔ رجیستری است که می‌خواهیم روی آن نوشته شود. ستون بعدی تعداد سیکل کلاک است که دستور استفاده می‌کند (۵). ستونهای سمت راست نیز شامل کدهای اسمبلی و توضیحات می‌باشند.

برنامه‌نویسی در C

روش دیگر برای نوشتن کد برای این تراشه‌های سیپرس استفاده از مفسر C و محیط ارتقاء آن است.

مزیت‌های C

در مقایسه با برنامه‌نویسی با زبان اسمبلی، استفاده از C چندین مزیت دارد.

- استاندارد بودن ـ اگر تجربه‌ای در برنامه‌نویسی C داشته باشید، با عبارتهای آن آشنایید و می‌توانید با سرعت بیشتری آغاز کنید. همچنین ممکن است بتوانید از کدهای C که برای تراشه‌های دیگر نوشته شده‌اند با تغییرات جزئی استفاده کنید.
- دستورات بیشترـ به جای استفاده از پرشهای ساده، کدهای شما می‌توانند از دستوراتی همچون if…else و case یا for و while … do استفاده کنند.
- اپراتورهای بیشترـ این مفسر از اپراتورهای ریاضی بیشتری پشتیبانی می‌کند و شما می‌توانید از جمع، تفریق، ضرب، تقسیم و مقایسه‌های گوناگون استفاده کنید.

- کتابخانه‌ها و مثالها ـ کتابخانه ها می‌توانند با استفاده از توابع معمولی مقدار زیادی در زمان، صرفه‌جویی کنند. کتاب‌خانه‌هایی برای برنامهٔ تراشه مدارهای واسط، میکروواير، I^2C و UART، زمانهای تأخیر، واسط صفحه کلید و LCD و توابع ریاضی وجود دارد. این مثالها شامل کدهای کامل برای صفحه کلید و ماوس می‌باشند.
- بهینه‌سازی ـ مفسر بهینه‌سازی‌ای به منظور کدها برای فشردگی و سرعت داراست. اما مشکل آنجاست که باید این مفسر را خریداری نمایید، در حالی که مترجم مجانی می‌باشد.

معماری تراشه

این تراشه ارزان قیمت با طراحی آسان است و به منظور استفاده در ابزارهایی که قصد انتقال بلاک‌های کوچک داده با سرعت متوسط، ساخته شده است و کاربردهای آن در وسایل جانبی استاندارد از قبیل ماوس یا دستگاه‌های نقطه‌یابی دیگر و واحدهای data-acquisition می‌باشد.

به عنوان مثال، واحدهای data-acquisition ممکن است نتایج خوانده شده از یک حسگر را به صورت متناوب به کامپیوتر بفرستد. پایه‌های I/O تراشه کنترلی می‌تواند به یک تبدیل کننده آنالوگ به دیجیتال که مقادیر خوانده شده از حسگر را به اعداد دیجیتالی تبدیل می‌کند و صل گردد. کامپیوتر میزبان نیز می‌تواند از اتصال USB برای درخواست آخرین داده‌های خوانده شده استفاده کند یا ممکن است کامپیوتر

سیگنال‌هایی را به رله‌ها، موتورها یا دستگاه‌های دیگری که پایه‌های کنترلی I/O تراشه به آن متصل هستند ارسال کند.

به جای تکرار مسائل موجود در کاتالوگ، به مطالب مهمی که قبل از کار با تراشه باید مورد نظر قرار گیرد توجه کنیم. نکات مشکل و گیج کننده کاتالوگ نیز مورد بحث قرار می‌گیرد.

خصوصیات و محدودیت‌ها

یکی از دلایل انتخاب تراشه ٦٣٧٤٣، ارزان قیمت بودن آن است. قیمت این تراشه حدود چند دلار در سفارشهای محدود می‌باشد.

تراشه دارای ۸ کیلوبایت حافظه برنامه است. با یک بهینه‌سازی، کدهایی که برای پشتبانی از ارتباطات USB لازم است، می‌توانند در یک کیلوبایت جای گیرند و به این ترتیب ۷ کیلوبایت باقیمانده می‌توانند برای کاربردهای دیگر استفاده شوند.

یک ابزار ضروری برای ارتقای این تراشه کیت ارتقا می‌باشد که شامل بر ارتقا، مترجم و برنامه‌های اشکال زدایی است. همچنین ممکن است احتیاج به برنامه‌ریز PROM Lo – CY3649 Hi نیز داشته باشید که همه این ابزارها توسط سیپرس در دسترس قرار گرفته است.

٦٣٧٤٣ برای همه پروژه‌ها مناسب نیست. این تراشه دارای سرعت پایین است که به معنای آن است که شما نمی‌توانید به منظور انتقالهای همزمان و توده‌ای از آن استفاده

کنید. و سریعترین زمان تأخیر ممکن دارای انتقال وقفه‌ای، ۸ بایت در هر ۱۰ میلی‌ثانیه می‌باشد. برخلاف بعضی از کنترلرهای اولیه، 63743 از انتقال وقفه‌ای خروجی پشتیبانی می‌کند.

درون تراشه

CPU این تراشه یک RISC هشت بیتی است که می‌تواند به حافظه برنامه، RAM، پورت‌های I/O همه کاره و البته پورت USB دسترسی داشته باشد. پورت USB در حقیقت یک پورت سوئیچ خودکار است که هر دو واسط USB و PS/2 را برای ماوس و دیگر دستگاه های نقطه‌یابی ممکن می‌سازد. این ویژگی به منظور طراحی دستگاه‌هایی که قابل تطبیق با هر دو باس باشند قرار گرفته است. وقفه‌ها و ریست‌های مختلفی می‌توانند به CPU وقفه بدهند.

حافظه

حافظه داخلی تراشه ٦٣٧٤٣، شامل هشت کیلوبایت (از ۰۰۰۰ h تا FFFh ۱) از نوع OTP PROM برای ذخیره برنامه و ۲۵٦ بایت RAM (از ۰۰h تا FFh) برای ذخیره داده‌های موقتی می‌باشد. ۳٤ بایت رجیستر I/O، هر کدام با وظیفه‌ای تعریف شده، نیز در این تراشه وجود دارد.

سازماندهی حافظه برنامه تراشه، شبیه به میکروکنترلرهای دیگر است. اجرای برنامه از آدرس h ۰۰ آغاز می‌شود. آدرس‌های h۰۰ و h ۰۱ حاوی آدرسی هستند که کد اصلی برنامه از آنجا شروع می‌گردد.

آدرس‌های h۰۲ تا h۱۷ حاوی اشاره‌گرهای وقفه‌ای هستند که وقتی یک از یازده وقفه تراشه اتفاق می‌افتد، آدرسی را که میکروکنترلر باید به آنجا پرش کند مشخص می‌نمایند. در اینجا مثالی از جدول اشاره‌گرهای برنامهٔ تراشه آمده است.

```
ORG     00h
jmp         reset       ;device reset
jmp         bus-reset   ; USB reset interrupt
jmp         error       ; 128- microsecond interrupt
jmp         lms-timer   ; 1.028- millisecond interrupt
jmp         endpoint0   ; Endpoint  0  interrupt
jmp         endpoint1   ; Endpoint  1  interruopt
jmp         endpoint2   ; Endpoint  2  interruopt
jmp         spi         ; SPI interrupt
jmp         capture-a   ; capture timer  A  interrupt
jmp         capture-b   ; capture timer  B  interrupt
jmp         gpio        ; GPIO interrupt
jmp         wakeup      ; Wake-up  interrupt
```

هر اشاره‌گر وقفه، به آدرسی که عملوند آن مشخص می‌کند پرش می‌نماید. وقفه‌هایی که استفاده نمی‌شوند، نباید اتفاق بیفتد اما برنامهٔ تراشه باید شامل پرشهایی برای این وقفه‌ها

باشد. زیر برنامه سرویس وقفه (ISR) برای این وقفه‌های بدون استفاده، بدون تغییر دادن رجیسترها فقط باید اجرای برنامه را به محل فراخوانی، بازگرداند.

وقفه‌ها بر اساس اولویت از آدرس ۰۰۰۲ h با بیشترین اولویت نوشته می‌شوند. حافظه برنامه از ۰۰۱۸h تا 1FDFh برای ذخیره بقیه کدها در دسترس می‌باشند.

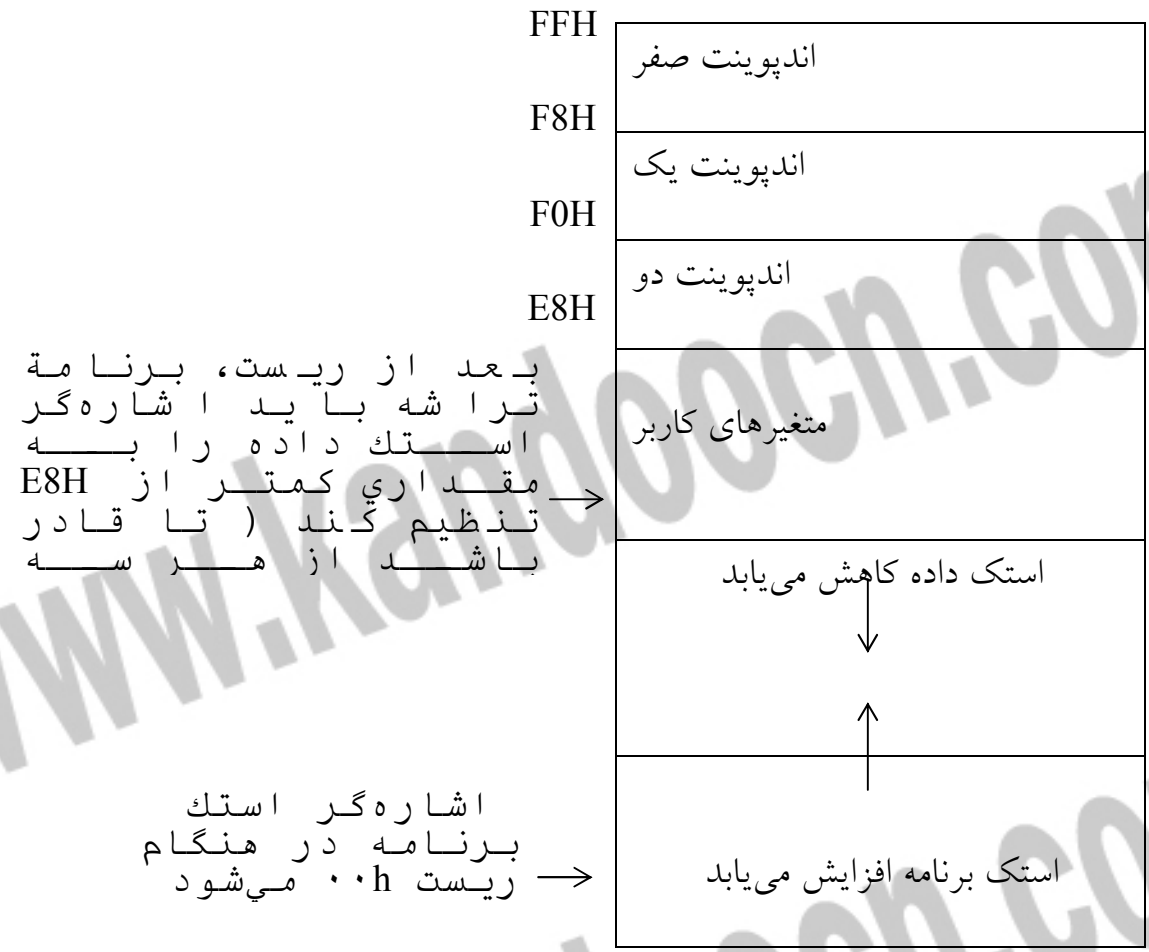
۲۵٦ بایت RAM باید دو پشته داده و ۸ بایت بافر برای اندپوینت، و داده‌های موقتی دیگری را نگهداری کنند (شکل ٤-۸). بافرهای اندپوینت از آدرس‌های E8h تا FFh استفاده می‌نماید.

پشته‌ها دارای ساختار LIFO (آخرین ورودی – اولین خروجی) هستند. RAM دارای دو اشاره‌گر برای دسترسی به دو پشته می‌باشد. اشاره‌گر پشته برنامه (PSP) از آدرس ۰۰ h در هنگام ریست شروع می‌شود و مقدارش زیاد می‌گردد. در حالی که اشاره‌گر پشته داده (DSP) ممکن است توسط سخت‌افزار به E8h یا کمتر تنظیم گردد و مقدارش کم می‌شود. برنامهٔ تراشه نیاز دارد که اطمینان حاصل کند که پشته‌ها آن قدر بزرگ نمی‌شوند که وارد محدودهٔ دیگری گردند.

اشاره‌گر پشته برنامه

اشاره‌گر پشته برنامه (PSP) آدرس‌هایی را کد پس از بازگشت از یک زیر برنامه فراخوانی شده یا سرویس وقفه باید به آنها بازگردد را نگهداری می‌کند. در وقفه‌ها، PSP همچنین حالت پرچمهای صفر و نقلی را ذخیره می‌کند. برنامهٔ تراشه مجبور نیست برای

رهبری PSP همهٔ کارها را خود انجام دهد. همهٔ این کارها بـه صـورت خودکـار توسـط سخت افزار و دستورات CALL و RET و RETI انجام می‌شود.



شکل ٤-٨: RAM تراشه enCoRo حاوی بافرهای اندپوینت‌ها، پشته‌های داده و برنامه و متغیرهایی می‌باشد که برنامهٔ تراشه نیاز دارد.

در هنگام ریست، PSP به آدرس ۰۰ h اشاره خواهد کرد. SPS قادر به سرویس دهی به چندین وقفه و یا زیربرنامه است. پس از اجرای زیربرنامه‌ها به دستوری که بعد از دستور فراخوانی است، باز خواهیم گشت. مثلاً اگر PSP قبل از فراخوانی یک زیربرنامه به آدرس ۰۰h در حافظه برنامه اشاره کند، دستور CALL همچنین PSP را دو واحد افزایش می‌دهد (به خانه ۰۲h اشاره خواهد کرد) بنابراین آماده است که در صورت لزوم آدرس دیگری را بگیرد. دستور RET که باعث بازگشت از زیر برنامه می‌شود، مقداری

را که توسط PSP نشان داده می‌شود به درون شمارنده برنامه بارگذاری کرده و PSP را دو واحد کم می‌نمای. سپس اجرای برنامه از محلی که فراخوانی از آنجا انجام شده لود ادامه می‌یابد.

نظیر همین مراحل در زیر برنامه سرویس وقفه اتفاق می‌افتد، فقط با این تفاوت که مقادیر پرچمهای صفر و نقلی نیز ذخیره می‌شوند.

اشاره‌گر پشته داده

اشاره‌گر پشته داده (DSP) داده‌هایی را که توسط دستور PUSH ذخیره می‌شوند، نگهداری می‌کند. مثلاً PUSH A محتویات آکومولاتور را در پشته داده ذخیره می‌کند. DSP پس از ذخیره یک بایت، یک واحد کاهش می‌یابد. دستور POP بایت‌هایی را که قبلاً ذخیره شده است را بازیابی می‌کند و DSP را یک واحد افزایش می‌دهد.

مقدار پیش فرض DSP در هنگام ریست جایی که باید باقی بماند نیست. غیر از تراشه‌هایی که اصلاً از USB استفاده نمی‌کنند، برنامهٔ تراشه باید قبل از استفاده از هر دستور PUSH ابتدا DSP را به مقدار جدید تنظیم کند. در هنگام ریست DSP مقدار ۰۰h را دارد. از اینجا، هر دستور PUSH باعث می‌شود که DSP کاهش یافته و به بالای RAM (FFh) برود که بایت ۷ بافر اندپوینت صفر است. به این دلیل، قبل از هر PUSH، برنامهٔ تراشه باید اشاره‌گر DSP را به E8h یا کمتر تنظیم کند.

; Store the DSP’s new beginning address

; in the accumulator.

mov A/ 70h

; Swap the contents of the accumulator with rhe DSP swap A/ dsp

ارتباطهای USB

برنامهٔ تراشه، موتور واسط سریال (SIE) را توسط دسترسی به رجیسترها کنترل می‌کند.

نه عدد رجیستر وجود دارند که به صورت مستقیم با ارتباطهای USB مرتبطند:رجیستر آدرس، سه رجیستر حالت اندپوینت، سه رجیستر شمارنده اندپوینت، یک رجیستر کنترل و وضعیت و رجیستر فعال‌ساز وقفه.

آدرس دستگاه

رجیستر آدرس دستگاه USB بیت آدرسی را که توسط میزبان در مرحله سرشماری به دستگاه نسبت داده شده است، نگهداری می‌کند. سخت افزار باید خواسته Set –Address را تشخیص داده، تأیید متقابلی در پاسخ به خواسته فرستاده و آدرس رسیده را در این رجیستر ذخیره نماید. بیت ۷ باید در ۱، ست شود تا موتور واسط قادر شود به ترافیک‌های USB پاسخ دهد.

حالت‌ها

رجیستر حالت اندپوینت صفر حاوی اطلاعاتی درباره آخرین پاکت داده رسیده به اندپوینت صفر است. SIE و برنامهٔ تراشه هر دو قادرند که محتویات این رجیستر را تغییر دهند.

سه بیت مشخصهٔ پاکت نوع پاکت توکن را مشخص می‌کنند: Setup، ورودی یا خروجی. در طول فاز داده از ترنزکشن Setup، SIE بیت تنظیم را یک می‌کند. برای جلوگیری از دوباره نوشته شدن، برنامهٔ تراشه در صورت یک بودن این بیت اجازه نمی‌دهد که هیچ عمل نوشتنی روی بافر USB انجام شود. برنامهٔ تراشه تا وقتی که همه بایت‌های داده دریافت نشود، نمی‌تواند این بیت را تغییر دهد.

بیت ACK نیز وقتی که ترنزکشن با موفقیت کامل شود، یک می‌گردد.

چهار بیت حالت چگونگی پاسخ SIE به ترنزکشن‌های Setup و ورودی و خروجی را مشخص می‌کنند. بسته به نوع ترنزکشن، برنامهٔ تراشه می‌تواند از SIE بخواهد که ACK، NAK، Stall یا پاکت داده‌ای با طول صفر بفرستد. در پاره‌ای موارد، SIE پس از ACK حالت را تغییر می‌دهد. مثلاً وقتی که حالت به صورت ACK خروجی است، پس از بازگرداندن ACK در پاسخ به داده‌های رسیده، SIE حالت را به Nak OUT تنظیم می‌کند. این مسأله به سخت‌افزار امکان می‌دهد که داده‌های رسیده‌ای را که با ACK پاسخ داده شده‌اند، بازیابی کند. پس از بازیابی این داده‌ها، برنامهٔ تراشه می‌تواند برای امکان دریافت داده‌های جدید بیت‌های حالت را به ACK OUT تغییر دهد.

درک نحوه استفاده از این بیت‌های حالت بسیار گیج کننده بود. سیپرس چهار صفحه در مورد چگونگی پاسخ به همه این رویدادها تهیه کرده است. خوب است این حالت‌ها را بر اساس اینکه چه اندپوینت‌هایی در چه وضعیتی از آنها استفاده می‌کنند، گروه‌بندی

کنیم. جدول ۳-۸ حالتهایی را که توسط اندپوینت صفر استفاده می‌شود نشان می‌دهد. در هر کدام از این حالتها همانند اندپوینت کنترلی، ترنزکشن‌های Setup پذیرفته می‌شوند.

مکمل رجیسترهای حالت اندپوینت صفر، رجیستر حالت اندپوینت یک و رجیستر حالت اندپوینت دو می‌باشند. این رجیسترها نیز همانند اندپوینت صفر دارای بیت‌های ACK و حالتهای مشابه می‌باشند. این رجیسترها بیت‌های مشخصهٔ پاکت ندارند چون فقط از انتقال‌های ورودی و خروجی پشتیبانی می‌کنند. هر کدام از این رجیسترها دارای بیت‌های استال نیز هستند.

اندپوینت‌های ۱ و ۲ از تنظیمات حالت متفاوتی با اندپوینت صفر استفاده می‌کنند چون این اندپوینت‌ها احتیاج به پاسخ دادن به پاکتهای Setup ندارند در حالی که اندپوینت صفر این وظیفه را دارا می‌باشد. جدول ٤-۸ حالتهایی را که توسط اندپوینت‌های ۱ و ۲ استفاده می‌شوند نشان می‌دهد. جدول همچنین چگونگی استفاده برنامهٔ تراشه از بیت استال که باعث می‌شود SIE در حالت‌های ACK IN و ACK OUT ، استال را بازگرداند.

جدول ۳-۸: حالت‌هایی که با اندپوینت صفر در رجیستر حالت مربوطه استفاده می‌شوند.

اندپوینت صفر باید ترنزکشن‌های Setup را قبول کند

حالت	رمزگذاری Setup	پاسخ به ترنزکشن			حالت بعد	کاربرد عمومی
		ورودی	خروجی	از ACK		
NAK ورودی خروجی	۰۰۰۱	پذیرش	NAK	NAK	مشابه	انتقالی در حال انجام نیست. منتظر ترنزکشن setup می‌باشد.
فقط وضعیت خروجی	۰۰۱۰	پذیرش	استال	بررسی	مشابه	انتقال کنترلی خواندن، مرحله وضعیت، ارسال ACK در قابل دریافت پاکت داده صفربایتی با زنجیره داده صحیح.
استال ورودی خروجی	۰۰۱۱	پذیرش	استال	استال	مشابه	انتقالی در حال انجام نیست. منتظر ترنزکشن setup بعدی است.
چشمپوشی ورودی خروجی	۰۱۰۰	پذیرش	چشمپوشی	چشمپوشی	مشابه	انتقالی در حال انجام نیست. منتظر ترنزکشن setup بعدی است.
فقط وضعیت ورودی	۰۱۱۰	پذیرش	داده صفر بایتی	استال	مشابه	انتقال کنترلی نوشتن مرحله وضعیت، برای یک ترنزکشن ورودی، پاکت داده صفربایتی بازگردانده می‌شود.
NAK خروجی وضعیت ورودی	۱۰۱۰	پذیرش	داده صفر بایتی	NAK	مشابه	انتقال کنترلی نوشتن مرحله وضعیت، برای یک ترنزکشن ورودی، پاکت داده صفربایتی بازگردانده می‌شود.
ACK خروجی- NAK ورودی	۱۰۱۱	پذیرش	NAK	ACK	NA ورودی/ خروجی	ترنزکشن کنترلی نوشتن مرحله داده.
NAK ورودی- وضعیت خروجی	۱۱۱۰	پذیرش	NAK	بررسی	مشابه	انتقال کنترلی خواندن، مرحله داده یا وضعیت برای یک

						ترنزکشن، ورودی، داده را باز می‌گرداند. برای یک ترنزکشن خروجی در پاسخ به دریافت پاکت داده صفر بایتی با بیت زنجیره داده صحیح، ACK باز می‌گرداند.
ACK ورودی – وضعیت خروجی	۱۱۱	پذیرش	داده	بررسی	NAK ورودی- وضعیت خروجی	انتقال کنترلی خواندن، مرحله داده یا وضعیت، برای یک ترنزکشن ورودی، داده را باز می‌گرداند. برای یک ترنزکشن خروجی در پاسخ به دریافت پاکت داده صفر بایتی با بیت زنجیره داده صحیح، ACK باز می‌گرداند.

جدول ٤-٨: حالتهایی که توسط اندپوینت ۱ و ۲ استفاده می شوند. اندپوینت‌هـای ۱ و ۲ ترنزکشن‌های Setup را قبول نمی‌کنند

حالت	رمزگذاری Setup	پاسخ به ترنزکشن			حالت بعد	کاربرد عمومی
		ورودی	خروجی	از ACK		
غیر فعال	۰۰۰۰	چشمپوشی	چشمپوشی	چشمپوشی	—	اندپوینت غیر فعال است
NAK خروجی	۱۰۰۰	چشمپوشی	چشمپوشی	چشمپوشی	—	اندپوینت خروجی آماده دریافت داده نیست
ACK خروجی (استال =۰)	۱۰۰۱	چشمپوشی	چشمپوشی	ACK	—	اندیوپنت خروجی ایست کرده است
ACK خروجی (استال =۰)		استال	چشمپوشی	چشمپوشی	—	اندپوینت خروجی ایست کرده است
NAK ورودی	۱۱۰۰	چشمپوشی	NAK	چشمپوشی	—	NAL ورودی اندپوینت ورودی داده‌ای برای فرستادن ندارد
ACK ورودی (استال =۰)	۱۱۰۱	چشمپوشی	داده	چشمپوشی	NAK ورودی	اندپوینت ورودی داده برای فرستادن دارد
ACK ورودی (استال =۱)		چشمپوشی	استال	چشمپوشی	—	اندپوینت ورودی ایست کرده است

کنترل و وضعیت اندپوینت

هر کدام از اندپوینت‌ها همچنین دارای یک رجیستر شمارنده اندپوینت هستند که حاوی اطلاعاتی دربارهٔ پاکت داده‌ای انتقال یافته یا در حال انتقال است. هر کدام از این رجیسترها دارای چهار بیت شمارنده، یک بیت زنجیره داده و بیت وجود داده، هستند. چهار بایت شمارنده، تعداد بایت‌های داده ترنزکشن را نگهداری می‌کند. در ترنزکشن ورودی، این مقدار مشخص می‌کند که چه تعداد بایت داده در ترنزکشن را نگهداری می‌کند. در ترنزکشن ورودی، این مقدار مشخص می‌کند که چه تعداد بایت داده در ترنزکشن بعدی فرستاده می‌شود، این تعداد بایت شامل بایت‌های CRC نمی‌شود. مقادیر مجاز بین صفر و هشت می‌باشند. در ترنزکشن‌های خروجی و Setup، این مقدار تعداد بایت‌هایی را که در آخرین ترنزکشن رسیده است مشخص می‌کند که این مقدار شامل دو بایت CRC نیز می‌شود. مقادیر مجاز بین ۲ تا ۱۰ می‌باشد. شمارنده خروجی و Setup تا هنگامی که برنامهٔ تراشه رجیستر را بخواند قفل می‌گردد.

در ترنزکشن‌های خروجی و Setup اگر مقادیر CRC رسیده صحیح نباشند، مقدار بیت وجود داده یک می‌گردد.

بیت Data-toggle حالت تغییر مشخصه پاکت داده را تعیین می‌کند. در ترنزکشن‌های ورودی، برنامهٔ تراشه این مقدار را تنظیم می‌کند و در ترنزکشن‌های خروجی و Setup این بیت را SIE تنظیم می‌نماید.

کنترل وضعیت USB

رجیستر کنترل وضعیت USB دارای دو بیت برای ارتباطات USB و چهار بیت برای ارتباط PS/2 یا USB و یک بیت برای ارتباط PS/2 می‌باشد. SIE بیت فعالیت باس را پس از تشخیص فعالیت بر روی باس یک می‌کند. برنامهٔ تراشه می‌تواند از این بیت برای تصمیم‌گیری در رفتن دستگاه به حالت بیکاری استفاده کند. اگر این بیت بیشتر از ۳ میلی ثانیه صفر باقی بماند، تراشه باید وارد حالت بیکاری شود.

بیت فعال ساز VREG قادر است که در خروجی VREG ولتاژ ۳/۳ V را فعال سازد. این خروجی برای مقاومت بالابر[1] USB به D- در باس است. چون VREG تحت کنترل برنامهٔ تراشه است، کد می‌تواند ولتاژ خروجی را برداشته یا حفظ کند تا اتصال یا جدا شدن دستگاه از باس را تشخیص دهد. امپدانس خروجی VREG حدود ۲۰ اهم است بنابراین مقدار مقاومت باید ۱/۳ K اهم باشد تا با ۱/۵ K مرجع خصوصیات سازگار شویم.

بیت حالت ریست USB – بیت مد وقفه فعال شدن PS/2، تعیین می‌کند که وقفه USB داده شود یا اینکه فعالیت PSP داشته باشیم.

سه بیت کنترلی، برنامهٔ تراشه را قادر می‌سازند که خطوط USB یا PS/2 را در وضعیت‌های خاصی تنظیم کنند، از جمله این وضعیت‌ها می‌توان از k،j و SE0 مربوط به

[1] - Pull-up

USB نام برد. اگر قبلاً میزبان قابلیت Remote-wakup را فعال کرده باشد، برنامهٔ کاربردی می‌تواند از وضعیت Force-k برای فرستادن سیگنال بازگشت که به میزبان می‌گوید دستگاه می‌خواهد ارتباط دوباره آغاز شود استفاده کند.

بیت فعال کردن PS/2 قادر است مقاومت بالابر داخلی که بین خطوط SDATA و SCLK است را برای استفاده ارتباط PS/2 فعال سازد.

رجیستر داده پورت ۲، حالت چهار بیت فقط خواندنی را در یک پورت ورودی کمیک نگهداری می‌کند. دو بیت، حالت +Dو -D در هنگام استفاده از USB یا حالت SCLK و SDATA در هنگام استفاده از PS/2 می‌باشند. دو بیت دیگر بیشتر مواقع می‌توانند به عنوان دو ورودی استفاده شوند. اگر مقاومت موجود بر روی خط -D از منبع ولتاژ خارجی برای راه‌اندازی استفاده کند و یا اینکه دستگاه از USB پشتیبانی نکند، از پایهٔ VREG می‌توان به عنوان ورودی استفاده کرد که در این حالت وضعیت این بیت از طریق P2.0 قابل دسترسی است.

وقتی که ساعت داخلی فعال است، مرجع زمانی بر روی پایهٔ XTALIN وجود نخواهد داشت و می‌توان از این پایه نیز به عنوان ورودی از طریق PS.1 استفاده کرد.

آخرین رجیستر مربوط به USB رجیستر فعال ساز وقفه‌های اندپوینت است، که وقفه‌ها را برای اندپوینت‌های صفر، ۱ و ۲ فعال می‌سازد. توضیحات مربوط به این رجیستر در زیر در بخش پردازش وقفه ارائه خواهد شد.

اصول راه‌انداز دستگاه

راه‌انداز دستگاه نرم‌افزاری است که برنامهٔ کاربردی را قادر می سازد که بـه سـخت‌افـزار دستگاه دسترسی یابد. بعضی از راه‌اندازهای دستگاه‌ها راه‌انداز کلاس هستند که می‌توانند با دستگاه‌هایی که کاربرد مشابه دارند ارتباط برقرار کنند.

رها کردن برنامه‌های کاربردی از جزئیات

یک راه‌اندا دستگاه، برنامـهٔ کـاربردی را از داشـتن جزئیـات دربـاره اتصـالهای فیزیکـی، سیگنال‌ها، پروتکل‌هایی که برای ارتباط با دستگاه لازم اسـت، بـی‌نیـاز مـی‌کنـد. برنامـهٔ کاربردی نرم‌افزاری است که کاربر آن را راه می‌اندازد که شامل پردازش کننده‌های word و داده‌ها تا برنامه‌هایی با کاربرد خاص کـه سـخت‌افـزار ویـژه‌ای را پشـتیبانی مـی‌کنـد، می‌شوند.

یک راه‌انداز دستگاه کدهای برنامهٔ کاربردی را قادر می‌سازد فقط بـا دانسـتن نـام وسـیلهٔ جانبی (مثل hp Laserjet) و یا کار آن (دستهٔ بازی) بتوانـد بـا آن ارتبـاط برقـرار کننـد. احتیاج نیست که برنامه کاربردی آدرس فیزیکی پورتی را که وسیلهٔ جانبی لـه آن متصـل شده بداند (مانند آدرس h ۳۷۸) یا اینکه سیگنال‌های تأیید

متقابل که به وسیله جانبی نیاز دارد (Busy. Strobe و غیره) را کنتـرل و نمـایش دهـد. برنامهٔ کاربردی حتی نمی داند که یک دستگاه از مدار واسطه USB استفاده مـی کنـد یـا

مدار واسط دیگری. با توجه به اینکه جزئیات سخت افزاری در یک زبان سطح پایین تر وجود دارد.

وظیفهٔ راه انداز دستگاه ترجمه ارتباط بین زبان سطح بالا برنامهٔ کاربردی به کدهای ویژه سخت افزار می باشد. برنامهٔ کاربردی از تابعهایی که سیستم عامل آنها را پشتیبانی می کند. برای ارتباط با راه انداز دستگاه استفاده می کند. کدهای ویژه سخت افزار نیز پروتکل لازم برای دسترسی به مدار وسیله جانبی را دارا هستند که شامل تشخیص حالت سیگنال های وضعیتی و استفاده از سیگنال های کنترلی در زمان لازم می باشد.

ویندوز دارای تابعهای کاربردی رابط با برنامه نویسان (API) است که برنامهٔ کاربردی را قادر می سازد که با راه انداز دستگاه ارتباط برقرار کند. برنامه هایی که در زبانهای برنامه نویسی ویژه وال بیسیک++C/C و دلفی نوشته می شوند قادر به فراخوانی تابعهای API هستند سه تابعی که راه انداز دستگاه را قادر می سازد که از دستگاه USB بخواند و بنویسد عبارتند از DeviceIoControl. Write File, ReadFile

به منظور اینکه برنامه نویسی آسانتر و صحیح تر باشد، ویژوال بیسیک برای کارهای معمولی دارای کنترل هایی است به عنوان مثال، برنامهٔ کاربردی می تواند با استفاده از موضوع Printer داده را به چاپگر و کنترل McComm بفرستد تا به دستگاهی که به پورت سریال RS-232 متصل است ارتباط برقرار کند. استفاده از این کنترل ها راه راحت تر و کم اشتباه تری برای تنظیم کردن پارامترها و تبادل اطلاعات می باشد. در پشت کد

های کنترلی ممکن است تابعهای API وجود داشته باشند که با راه انداز از دستگاه ارتباط برقرار می کنند. اما موضوعهای کنترلی برنامه نویسان را از سرو کار داشتن با آنها راحت می کند.

ویژوال بیسیک موضوع کنترلی عمومی برای ارتباط با USB ندارد. نحوهٔ برقراری ارتباط برنامهٔ کاربردی با دستگاه USB بسته به راه انداز آن متفاوت خواهد بود. به عنوان نمونه، یک برنامهٔ کاربردی ویژوال بیسیک می تواند از موضوع چاپگر برای ارتباط با چاپگر USB استفاده کند.

بعضی از راه اندازهای دستگاهها، راه اندازهای یکپارچه هستند که همهٔ کارها از ارتباط با برنامهٔ کاربردی گرفته تا خواندن و نوشتن به روی پورت یا آدرس ها ی حافظه که به سخت افزار دستگا متصلند، را به تنهایی انجام می دهند.

بعضی دیگر، که راه اندازهای ویندوز برای دستگاههای USB از این دسته اند. از مدل راه انداز از لایه ای استفاده می کنند که هر راه انداز یک بخش از وظیفه ارتباط را انجام می دهد. بالاترن لایه دارای راه انداز کاربردی است که ارتباط بین برنامهٔ کاربردی و راه انداز باس را که زبان سطح پایین تری دارد، رهبری می کند در لایه پایین تر راه اندا باس وجود دارد که ارتباط بین راه انداز کاربردی و سخت افزار، را رهبری می نماید. ممکن است یک یا چند راه انداز دیگر نیز ارتباط راه اندازی های کاربردی و باس را کاملتر کنند.

به طور کلی راه اندازهای لایه ای پیچیدگی بیشتری دارند اما لایه ای بودن باعث راحت تر شدن نوشتن راه اندازها می شود. دستگاهها ممکن است برای کارهای مشترکی که دارند، کدهای مشترکی داشته باشند. به علاوه، راه اندازهایی که ارتباط سخت افزار سیستم USB و ورودی ویندوز را برقرار می کنند، نویسندگان راه انداز را از تهیه مجدد آنها بی نیاز می کنند. نوشتن یک راه انداز دستگاه USB بسیار راحت تر از نوشتن راه اندازی است که همه جزئیات دسترسی به سخت افزار را دارا باشد

انواع استاندارد دستگاهها

بسیاری از وسایل جانبی، سازگار با کلاس های استاندارد هستند مانند، راه اندازهای دیسک، چاپگر ها، مودم ها، صفحه کلید ها و ماوس ها. همهٔ این دستگاهها از طریق مدارهای واسط مختلف قابل دسترسی می باشند که شامل USB نیز می شود. به عنوان مثال، یک صفحه کلید ممکن است از مدارهای واسط قدیمی یا USB استفاده کند یا یک راه انداز دیسک می تواند از مدارهای واسط متفاوت از جمله SCST,ATAPI، پورت چاپگر، IEEE-1394 یا USB استفاده کند.

ویندوز برای انواع دستگاههای استاندارد دارای راه اندازهایی با نام راه اندازهای کلاس است وقتی که دستگاههای یک کلاس ممکن است مدارهای واسط متفاوتی داشته باشند، راه اندازهای مکملی می توانند خصوصیات مختلف مدار واسط را پشتیبانی کنند. اگر

دستگاه دارای قابلیتهای بیشتر از راه انداز کلاس است، یک راه انداز فیلتر ویـژه دسـتگاه می تواند آنها را در صورت لزوم پشتیبانی کند.

دستگاههای شخصی

بعد از وسایل جانبی، دستگاههای شخصی هستند که به منظـور اسـتفاده در برنامـه هـای کاربردی ویژه ای طراحی می شوند مانند واحـدهای اکتسـاب داده، کنتـرل کننـده هـای موتور و وسایل تست ویندوز هیچ آگاهی از این دستگاهها نـدارد و بـه ایـن ترتیـب راه اندازهای خاصی نیز برای آنها نخواهد داشـت. دسـتگاههایی شـبیه بـه اینهـا بایـد از راه اندازهای شخصی استفاده کنند و یا آنکه طوری شوند که قابل تطبیق بـا کـلاس خاصـی باشند.

راه انداز دستگاه

در یک تعریف کلی، راه اندازی دستگاه یک سری کـد اسـت کـه جزئیـات ارتبـاط بـین سخت افزار دستگاه و CPU را به عهده دارد. حتی ممکن است یک زیر برنامـه کوچـک در برنامهٔ کاربردی نقش راه انداز از دستگاه را داشته باشـد. در ویـنـدوز کـدها بـرای راه انداز ها که شامل راه انداز های USB نیز می شود. از کدهای برنامـهٔ کـاربردی متفـاوت هستند، چون سیستم عامل کدهای راه انداز را با زبان سطح پـایین تـری از برنامـه هـای کاربردی اجازه می دهد.

مدهای کابر و هسته

در ویندوز، کدها با یکی از دو مد زیر راه اندازی می شوند: کاربر یا هسته. که هر کدام امتیازهای مختلفی را برای دسترسی به حافظه و دیگر منابع سیستم ایجاد می کنند. برنامهٔ کاربری باید در مد کاربر راه اندازی شود. بیشتر راه اندازها، از جمله راه اندازهای USB باید در مد هسته راه اندازی شوند، با این وجود که دستگاههای USB ممکن است راه اندازهای مکمل مد کار بر نیز داشته باشند.

در مدر کاربر، ویندوز دسترسی به حافظه و دیگر منابع را محدود می کند. ویندوز اجازه نمی دهد که برنامهٔ کاربردی به فضا حافظه ای که سیستم عامل برای حفاظت از آن طراحی شده است، دسترسی داشته باشد. این موضوع کامپیوتر را قادر می سازد که چندین برنامه را در یک زمان راه اندازی کند، بدون اینکه برای هم مزاحمت ایجاد کنند. در تئوری اگر یک برنامهٔ کاربردی خراب شود روی برنامه های دیگر اثر نمی گذارد. البته در حقیقت این موضوع همیشه صحیح نیست. در پنتیوم و پروسسورهای X86، مد کاربر مربوط به مد Ring3 پردازنده است.

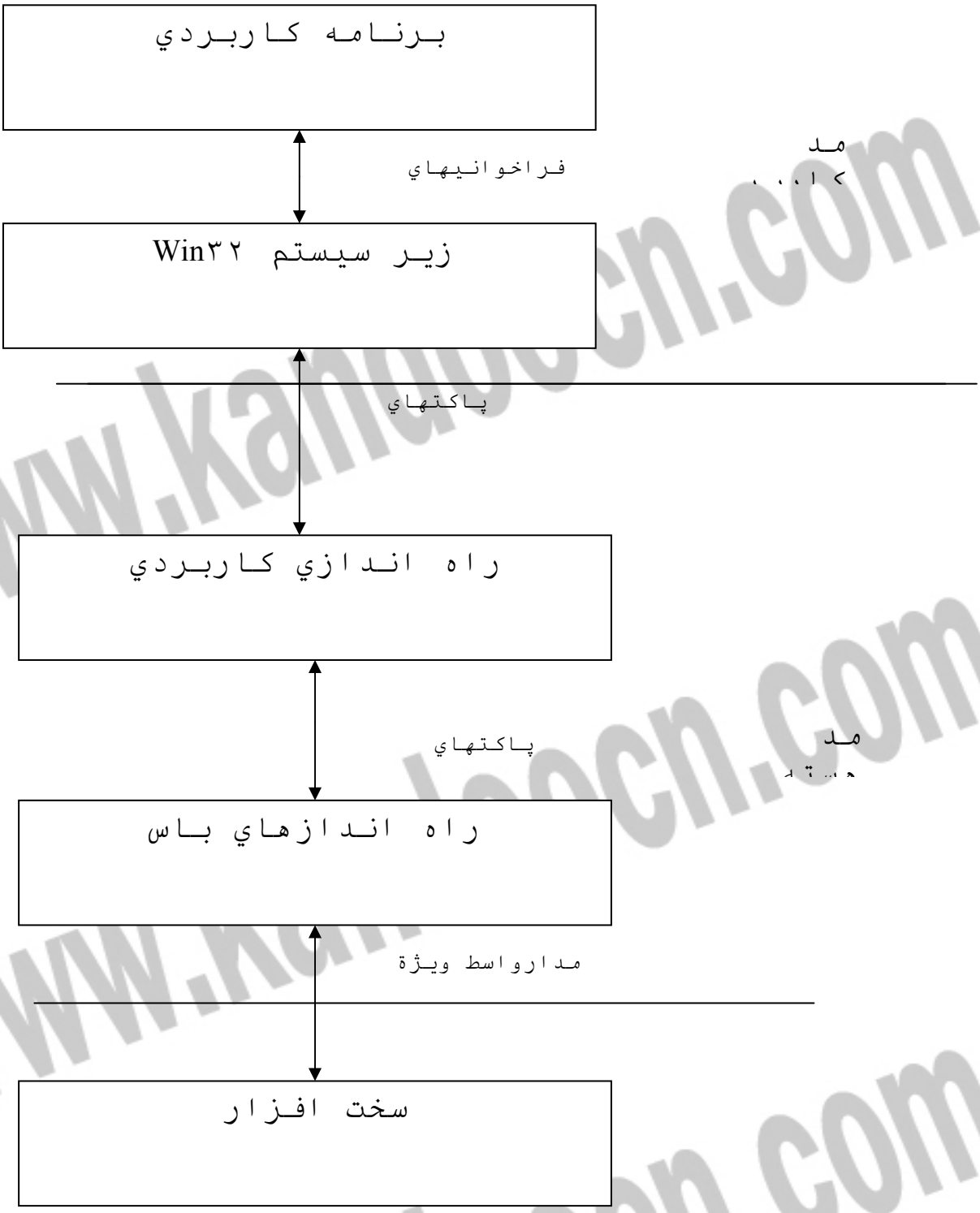
در مد هسته ای کدها در دسترسی به منابع سیستم، که شامل توانایی اجرای رهبری حافظه و کنترل دسترسی به پورت های I/O می شود محدودیت ندارند. در پنتیوم و دیگر پردازنده های X86، مد هسته ای مربوط به مد Ring0 پردازنده می باشد.

در ویندوز ۹۸ و Me برنامه های کاربردی می توانند به طور مستقیم و بدون راه اندازهای سطح پایین به پورت های I/o را دارند.

برنامه های کاربردی و راه اندازها هر کدام از زبان خاص خود برای ارتباط با سیستم عامل استفاده می کنند. برنامهٔ کاربردی از توابع Win32API بهره می برد و راه اندازها برای برقراری ارتباط با یکدیگر از فراخوانی ساختاری با نام پاکتهای خواسته I/O (IRPs)[1] استفاده می کنند.

ویندوز یک مجموعه از IRP را تعریف می کند که هر راه اندازی آنها را به کار برد. هر IRP یک عمل ورودی یا خروجی از خواستار می شود. یک راه انداز کاربردی برای دستگاه USB از IRP برای انتقال ارتباط از یا به راه انداز باس استفاده می کند. راه اندازهای باس درون ویندوز موجود هستند و احتیاجی نیست توسط برنامه نویسان برنامه های کاربردی یا نویستندگان راه انداز دستگاه طراحی شوند.

[1]-I/O Request Packets



شکل ۱-۱۰, USB از مدل راه اندازهای لایه ای تحت ویندوز با راه اندازهای مجزا برای دستگاهها و باس ها استفاده می کند.

مدل راه انداز Win32

راه اندازها دستگاه USB تحت ویندوز باید با مدل راه اندازی Win32 مطابقت کنند که توسط مایکرو سافت برای ویندوز ۹۸ و بالاتر که شامل ویندوز ۲۰۰۰ و Me می شود، تعریف شده است. این راه اندازها با نام راه اندازهای WDM شناخته می شوند که دارای حروف توسعه sys هستند(فایل های نوع دیگر نیز ممکن است از sys استفاده کنند).

مانند راه اندازهای سطح پایین دیگر، راه اندازهای WDM دارای تواناییهایی هستند که برنامه های کاربردی ندارند به این دلیل که راه انداز از طریق زبان سطح پایین تر با امتیاز های بیشتر باسیستم عامل ارتباط برقرار می کند. یک راه انداز WDM این توانایی را دارد که به یک برنامه کاربردی اجازه دسترسی به دستگاه را بدهد و برنامه دیگر این اجازه را ندهد. به عنوان مثال، راه انداز دسته بازی ممکن است اجازه استفاده در همه برنامه های کاربردی را داشته باشد یا اینکه فقط یک برنامه کاربردی امکان استفاده از آن را داشته باشد. ویژگی دیگری که ویندوز برای راه اندازهای WDM قرار داده است انتقال DMA و پاسخ به وقفه های سخت افزاری است.

مدل های راه انداز در ویندوزهای مختلف

مدل راه اندازی Win32 یک مدل راه اندازی معمولی را برای همه دستگاههای تحت ویندوز ۹۸ و یا بالاترفراهم می کند. نسخه های قدیمی تر ویندوز از مدل های متفاوتی برای راه اندازهای دستگاه استفاده می کردند. ویندوز ۹۵ از VxD ها (virtual device drivers) استفاده می کرد. ویندوز NT4 از نوعی راه انداز بهره می برد که راه اندازهای مد هسته ای نامیده می شدند. طراحانی که می خواستند هر دو ویندوز ۹۵ و NT را پشتیبانی کنند مجبور بودند که برای هر کدام راه انداز مجزایی را تهیه کنند. اما یک راه انداز WDM می تواند تحت ویندوز ۹۸ و ویندوز ۲۰۰۰ کار کند.

راه اندازهای باس USB که درون ویندوز است از نوع WDM هستند. با اینکه در ویندوز ۹۸ پشتیبانی از (Vx1 ها ادامه یافته است اما دستگاههای USB باید از راه اندازهای کاربردی WDM استفاده کنند چون این راه اندازهای کاربردی باید با راه اندازهای باس WDM ارتباط برقرار کنند.

مدل راه اندازی Win32 کاملا جدید نیست. اساس یک راه انداز WDM راه انداز مد هسته ای مربوط به NT به همراه ویژگیهای Plug-and-Play و مدیریت توان ویندوز ۹۵ می باشد. آخرین ویرایش ویندوز ۹۸ (نسخه OSR2.1 و بالاتر) تا حدی راه اندازهای WDM را پشتیبانی می کرد. این ویرایش به صورت جزئی در اختیار خریداران نبود اما فقط فروشندگانی که روی کامپیوترهای فروشی خود، نرم افزار نصب می کردند به آن

دسترسی داشتند. همراه با آمدن ویندوز ۹۸ پشتیبانی از WDM ها افزایش و پیشرفت کرد. چگونه ممکن بود که دو سیستم عاملی که قبلا احتیاج به راه اندازهای متفاوت داشتند اکنون از یک راه انداز استفاده کنند؟ ویندوز ۹۸ دارای راه اندازی به نام ntKernvxd بود که توسط آن راه اندازهای WDM تصور می کردند که با سیستم عاملی شبیه NT ارتباط برقرار می کنند. همه راه اندازهای WDM که در ویندوز ۹۸ کار می کردند احتیاج به این راه انداز دارند.

زبانهای برنامه نویسی

برنامه نویسان برنامه های کاربردی می توانند از زبانهای برنامه نویسی همچون ویژوال بیسیک، دلفی یا ویژوال ++) استفاده کنند. اما برای نوشتن یک راه انداز برای دستگاه USB شما احتیاج به ابزاری دارید که قابلیت کامپایل راه انداز WDM را داشته باشد و این به معنای استفاده از ویژوال C++ است. تنها استثناء جعبه های ابزار راه انداز هستند که یک راه انداز عمومی را تهیه می کنند و احتیاجی به برنامه نویسی ندارند یا به شما اجاز می دهند که از مفسرهای دیگر C یا دلفی برای استفاده اختصاصی از یک راه انداز عمومی استفاده کنید.

راه اندازهای لایه ای

در مدل راه اندازهای لایه ای که برای ارتباط با USB استفاده می شوند، هر لایه انجام بخشی از پروسه ارتباط را به عهده می گیرد. تقسیم کردن ارتباط به لایه های مختلف کار

مؤثری خواهد بود چون دستگاههای متفاوت دارای یک سری وظایف مشترک هستند که می توانند برای انجام آنها از یک نرم افزار مشابه استفاده کنند. به عنوان مثال، همه انواع دستگاهها ممکن است از USB استفاده کنند. پس بهتر است که یک مجموعه از راه اندازها برای کار با ارتباطهای ویژه USB که برای همه آنها مشترک است. تولید شود. قرارگرفتن این راه اندازها در ویندوز به این معنی خواهد بود که دیگر فروشندگان احتیاج به تهیه کردن آن ندارند.

لایه های راه انداز USB

قسمتی از ویندوز که ارتباط با دستگاه را راهبری می کند. زیر سیستم I/I/O است. زیر سیستم دارای چندین لایه است که هر کدام یک یا چند راه انداز برای کارهای مرتبط با خود دارند. خواسته ها از یک لایه به لایه بعد با توالی ارسال می شوند. یکی از قسمتهایی که در زیر سیستم I/O قرار دارد، زیر سیستم USB است که دارای راه اندازهایی است که ارتباطهای ویژه USB مربوط به همه دستگاهها را رهبری می کند.

مجموعه ای از پروتکل ها که توسط راه اندازها استفاده می شود. پشته نامیده می شود. شما می توانید لایه ها را به صورت پشته هایی که یکی روی دیگر است تصور کنید. برنامه های کاربردی، بالاترین پشته است و سخت افزار USB پایین ترین پشته خواهد بود.

راه انداز کاربردی

یک راه انداز کاربردی، برنامه کاربردی را قادر می سازد تا توسط تابعهای API با دستگاه USB صحبت کند. تابعهای API بخشی از زیرسیستم Win32 ویندوز هستند که مسئولیت تابعهای کاربر مانند راه اندازی برنامه کاربردی، رهبری ورودی کاربر از طریق صفحه کلید و ماوس و خروجیهای نمایشی روی صفحه نمایش را نیز به عهده دارد. برای ارتباط با دستگاه USB برنامه کاربردی احتیاجی ندارد که همه چیز را درباره پروتکل USB بداند.

راه انداز کاربردی همچنین از نحوه ارتباط با راه اندازهای سطح پایین که سخت افزار را کنترل می کنند. آگاه است شکل ۲-۱۰ چگونگی این کارها را در ارتباط با USB نشان می دهد. عموما از راه انداز کاربردی با لفظ راه انداز دستگاه یاد می شود با توجه به اینکه یک راه انداز دستگاه کامل با شامل بودن هر دو راه انداز باس و کاربردی شکل می گیرد. راه انداز کاربردی ممکن است راه انداز کلاس باشد یا راه انداز ویژه دستگاه وقتی که یک دستگاه یا زیر کلاس احتیاج به خصوصیاتی بیشتر از آن چیزی که یک راه انداز کلاس انجام می دهد دارد راه اندازهای مکملی که راه اندازهای فیلتر نامیده می شوند این قابلیتها را به آن اضافه خواهند کرد. یک راه انداز فیلتر فوقانی در بالای راه انداز کلاس قرار می گیرد و خواسته های برنامه کاربردی قبل از رسیدن به راه اندازکلاس به این راه انداز ارسال می شود. یک راه انداز فیلتر تحتانی نیز بین راه انداز کلاس و راه

انداز باس مقیم می شود راه انداز کلاس ممکن است خواسته های خود را به راه انداز فیلتر تحتانی بفرستد که آن هم آنها را به سمت راه انداز باس عبور می دهد. یک راه انداز فیلتر تحتانی می تواند یک راه انداز کلاس را قادر سازد تا مدارهای واسط مختلفی را پشتیبانی کند. به عنوان مثال، ویندوز راه اندازی دارد که راه انداز کلاس HID را قادر می سازد که با راه انداز باس USB ارتباط برقرار کند.

راه اندازهای باس

یک راه انداز باس USB از راه انداز هاب ریشه، راه انداز کلاس باس و راه انداز کنترلر میزبان استفاده می کند.

راه انداز هاب ریشه شناسایی کردن پورت ها و به طور کلی ارتباط بین راه اندازهای دستگاه و راه انداز کلاس باس را به عهده دارد. راه انداز کلاس باس سرشماری، انرژی باس، ترنزکشن های USB و ارتباط بین راه انداز هاب ریشه و راه انداز کنترلر میزبان را در دست دارد. راه انداز کنترلر میزبان، سخت افزار کنترلر میزبان را قادر می سازد که با نرم افزار سیستم USB ارتباط برقرار کند. کنترلر میزبان به باس متصل می شود. راه انداز کنترلر میزبان از راه انداز کلاس – باس مجزاست چون ویندوز از کنترل کننده های میزبان متفاوتی پشتیبانی می کند که هر کدام راه انداز مخصوص به خود را دارند.

راه اندازهای باس بخشی از ویندوز هستند و نویسندگان برنامه های کاربردی و راه اندازهای دستگاه احتیاجی به دانستن جزئیات نحوه کار آنها ندارند و در نتیجه

مایکروسافت مطالب کمی دربارهٔ آنها تهیه کرده است. اگر می خواهید بیشتر از طرز کـار این زبانهای سطح پایین آگاه شوید یک منبـع اطلاعـاتی کـدها و مقـالات و پـروژه‌هـای USB لینوکس است.