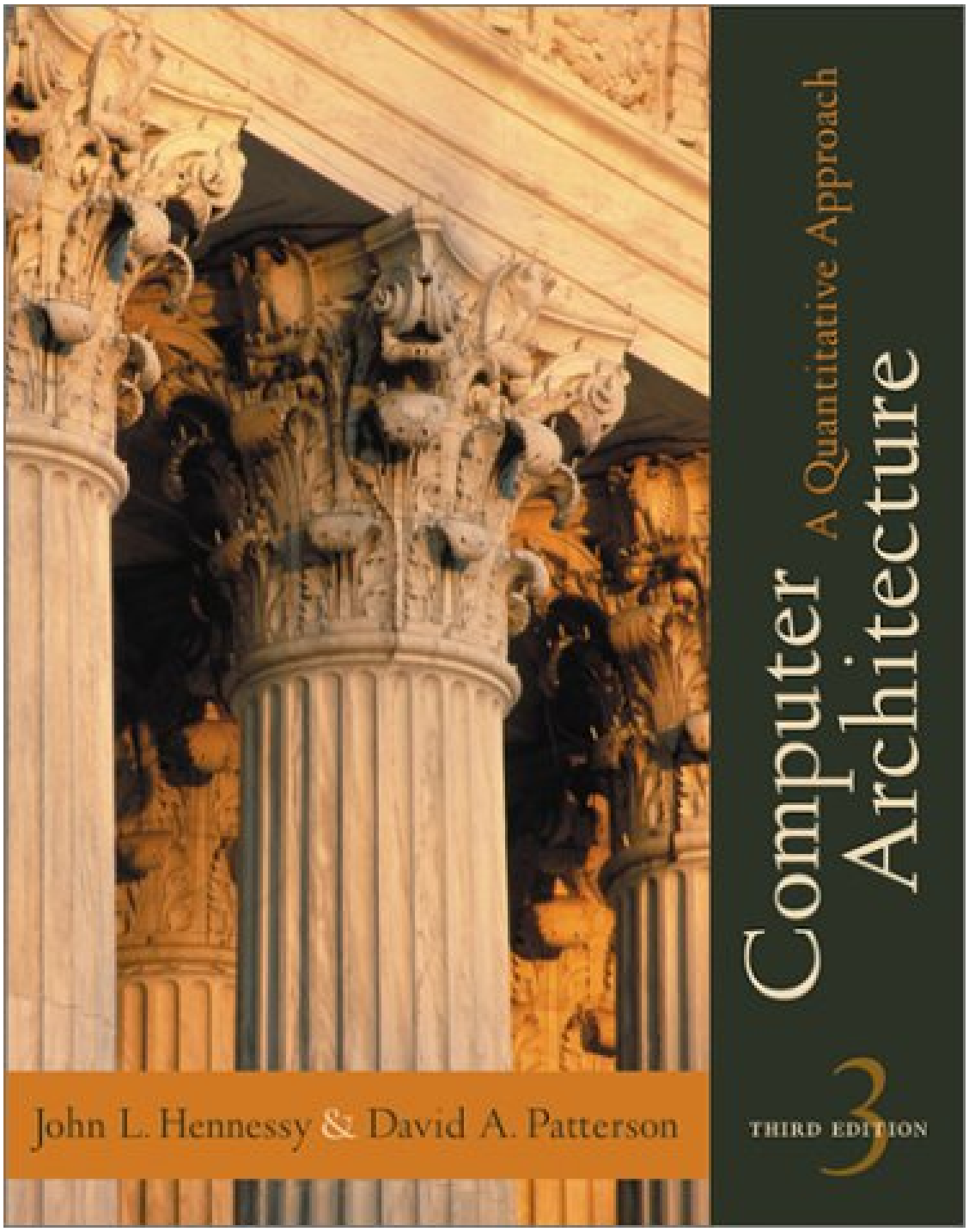
بسمه تعالی

# معماری سیستمهای کامپیوتری

**(Computer System Architecture)**

**مترجم: دکتر محمد حسن شیرعلی شهرضا**



# فصل اول : اصول طراحی کامپیوتر

## ۱ – ۱ مقدمه

فناوری کامپیوتر در ۵۵ سال اخیر از زمان ساخت اولین کامپیوتر الکترونیکی همه منظوره تاکنون پیشرفت چشمگیری داشته است. امروزه با کمتر از هزار دلار می‌توان یک کامپیوتر شخصی خرید که کارآیی آن، حافظه اصلی و حافظه دیسک آن از یک کامپیوتر یک میلیون دلاری سال ۱۹۸۰ بیشتر باشد. این رشد سریع به دلیل پیشرفت در اجزای سازنده کامپیوتر و توسعه در طراحی کامپیوتر می‌باشد.

گرچه توسعه فناوری کاملاً آشکار می‌باشد ولی به روشهای بهتر معماری کامپیوتر کمتر توجه می‌شود. در ۲۵ سال اولیه عمر کامپیوترهای الکترونیکی هر دو نیروی فوق نقش اصلی داشتند ولی از سال ۱۹۷۰ طراحان کامپیوتر وابستگی زیادی به مدارات مجتمع پیدا کردند.

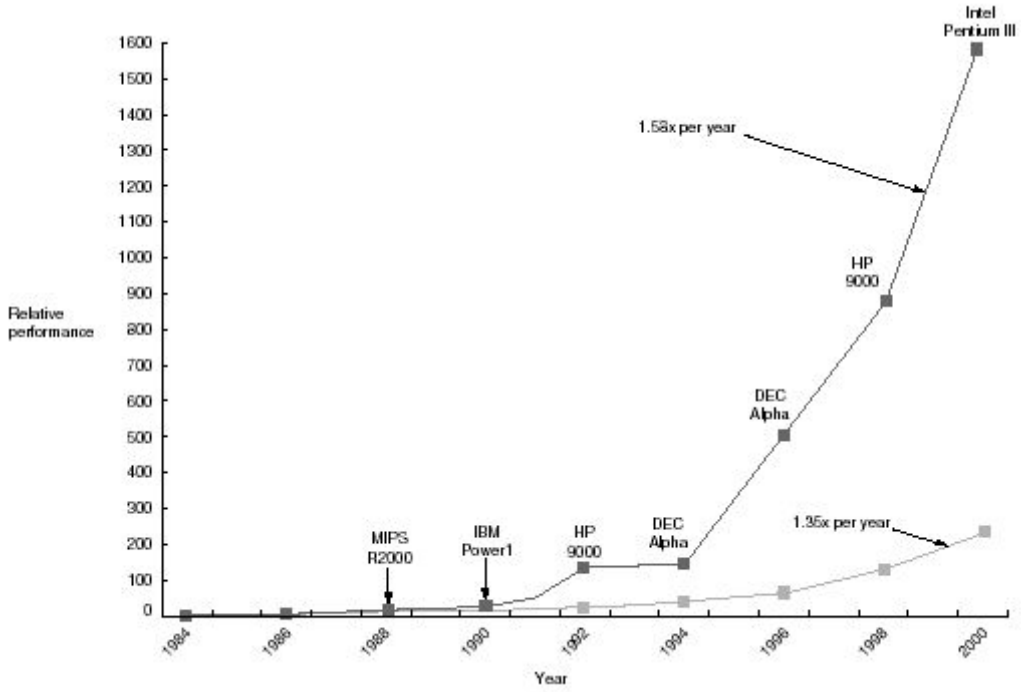
در سالهای حدود ۱۹۷۰ هر سال حدود ۲۵٪ تا ۳۰٪ در سال کارآیی کامپیوترهای Mainframe و مینی کامپیوترها افزایش می‌یافت. در سال ۱۹۷۰ ریزپردازنده ساخته شد. امکان ساخت ریزپردازنده بصورت یک مدار مجتمع باعث شد که کارآیی کامپیوترهای بزرگ و مینی کامپیوتر حدود ۳۵٪ در سال افزایش یابد.

این رشد سریع با قیمت مناسب ریزپردازنده‌های تولید انبوه ترکیب شده و باعث شد که درصد زیادی از تجارت کامپیوتر مربوط به ریزپردازنده‌ها باشد. دو عامل دیگر باعث شد که کامپیوتر در تجارت موفق‌تر باشد. یکی اجرای مجازی برنامه‌های اسمبلی که نیاز به همساز بودن کدها را کاهش داد و دیگری تولید سیستم‌های عامل استاندارد و مستقل از کمپانی خاص مانند یونیکس و اجزای آن مانند لینوکس که هزینه و ریسک تولید یک معماری جدید را کاهش داد.

این تغییرات باعث شد که امکان تولید معماری جدیدی که کامپیوتر کم دستور RISC مخفف (Reduced Instruction Set Computer) بود در سال ۱۹۸۰ فراهم شود. کامپیوترهای مبتنی بر RISC توجه طراحان را به دو نکته جلب کردند یکی موازی سازی در سطح دستورالعمل ILP = Instruction level parallelism که این کار ابتدا با خط لوله (Pipeline) و سپس با اجرای چند دستور همزمان انجام شد و دیگری حافظه نهان (Cache)، که در ابتدا ساختار ساده‌ای داشت ولی به مرور روشهای سازماندهی و بهینه سازی بهتری برای آن پیشنهاد شد. ترکیب معماری و سازماندهی باعث شد که در بیست سال بعدی هر سال حدود۵۰٪ رشد در کارآیی کامپیوترها بوجود آید. شکل شماره یک تفاوت نرخ رشد کارآیی را نشان می‌دهد.

این رشد سریع دو اثر داشت یکی در دسترس قرار گرفتن پیشرفتها برای کاربران، برای بسیاری از کاربردها یک ریزپردازنده با کارآیی بالای امروزی از یک سوپر کامپیوتر ۱۰ سال قبل قوی‌تر می‌باشد.

دومین اثر این پیشرفت، استفاده از کامپیوترهای مبتنی بر ریزپردازنده در تمام طراحیهای کامپیوتری می‌باشد. ایستگاه‌های کاری(Workstations) و کامپیوترهای شخصی(PC) تولیدات اصلی صنعت کامپیوتر می‌باشند. مینی کامپیوترها که قبلاً با مدارات معمولی یا آرایه‌های گیت ساخته می‌شدند با سرورهایی که از ریزپردازنده استفاده می‌کنند جایگزین شدند. کامپیوترهای بزرگ (Mainframe) بطور کامل با چند پردازنده‌هایی که از چندین ریزپردازنده معمولی استفاده می‌کنند جایگزین شدند. حتی سوپرکامپیوترها نیز امروزه با ترکیب ریزپردازنده‌ها ساخته می‌شوند.




شکل ۱

رشد کارآیی ریزپردازنده‌ها از اواسط ۱۹۸۰ میلادی تاکنون با توجه به معیار کارآیی SPECint

این نمودار با توجه به معیار کارآیی SPECint تهیه شده که در آن کامپیوتر VAX 11/780 معادل۱ فرض می‌شود. چون نمونه‌های مختلفی ازSPEC وجود دارد برای پردازنده‌های جدیدتر دو نمونه مختلف آن بررسی شده است مثلاً SPEC93 و SPEC95 قبل از اواسط دهۀ ۸۰ پیشرفت وابسته به فناوری بود و سالیانه حدود ۳۵٪ رشد می‌کرد. ولی بعد از آن پیشرفت وابسته به ایده‌های معماری و سازماندهی بود. در سال ۲۰۰۱ این پیشرفت به فاکتور ۱,۵ رسید. کارآیی محاسبات ممیز شناور حتی بیشتر از نمودار فوق می‌باشد

عدم وابستگی به همساز بودن با طراحیهای قدیمیو استفاده از ریزپردازنده باعث شد که حداکثر استفاده از فناوری امکان پذیر باشد. همین مسأله باعث رشد سریعتر مشاهده شده در شکل یک می‌باشد. کارآیی بدست آمده در سال۲۰۰۱ میلادی، ۱۵ برابر آنچه پیش‌بینی می‌شد، است.

در چند سال اخیر پیشرفت در مدارات مجتمع باعث شده که معماریهای قدیمی‌مانند X86 (IA-32)، از امکانات پیشرفته کامپیوترهای RISC بهره بگیرند. ساختارهای جدید X86 دستورات ساده و قدیمی‌را

توسط یک خط لوله شبیه RISC واکشی و اجرا می‌کنند. از سال ۱۹۹۰ تا کنون هزینه بالاسری ترانزیستورهای مورد نیاز برای ساختار X86 در مقایسه با ترانزیستورهای موجود در یک مدار مجتمع قابل صرف نظر کردن می‌باشد.

این کتاب دربارهٔ ایده‌های معماری و پیشرفت‌های کامپایلرها می‌باشد که این رشد سریع را امکان پذیر کرده‌اند. در قلب این رشد سریع ارائه روشهای کمّی(Quantitative) برای آنالیز و طراحی کامپیوترها بود. این مسأله نگاه اصلی این کتاب می‌باشد. پیشرفت‌های اخیر در کارآیی و کاهش قیمت کامپیوتر نیاز به پیشرفت‌های بیشتر در طراحی کامپیوتر را اجتناب ناپذیر می‌کند. و این پیشرفت‌ها در طراحی کامپیوتر با روشهای کمی‌در این کتاب بیان شده است. هدف این کتاب مستند کردن این روش و آشنایی شما با آن می‌باشد.

## ۱ – ۲ تغییر چهرهٔ محاسبات و وظیفه طراحان کامپیوتر

در دهۀ ۱۹۶۰ دنیای کامپیوتر در اختیار کامپیوترهای بزرگ(Mainframe) بود که میلیون‌ها دلار قیمت داشه، در اطاق کامپیوتر قرار گرفته و چندین اپراتور بر کار آنها نظارت داشتند. کاربرد عمده آنها پردازش داده‌های تجاری و محاسبات پیچیده عملی بود. در دهۀ ۱۹۷۰ مینی کامپیوترها متولد شدند، این ماشین‌ها که اندازه کوچکتری داشتند ابتدا برای آزمایشگاههای علمی‌ساخته شدند ولی به زودی این سیستم‌ها از طریق تقسیم زمانی (time – sharing) توسط چندین کاربر که بصورت محاوره‌ای(interactive) از طریق ترمینال‌های مختلف از یک سیستم استفاده می‌کردند، گسترش یافتند.

در دهۀ ۱۹۸۰ شاهد رشد کامپیوترهای رومیزی(desktop) مبتنی بر ریزپردازنده‌ها بودیم. این کامپیوترها جایگزین کامپیوترهای چندکاربره قبلی شدند و شاهد رشد سرویس‌دهنده‌ها(server) بودیم.

سرویس‌دهنده‌ها دارای حافظه و فضای دیسک و قدرت محاسبات بیشتری بوده و معمولاً از قابلیت اطمینان بالایی برخوردار هستند. در دهۀ ۱۹۹۰ شاهد ظهور اینترنت و صفحات شبکه‌ای جهان گستر(world wide web) بودیم. همچنین اولین وسایل محاسبات دستی(Personal Digital assistants) یا PDA و همچنین وسایل مصرفی الکترونیکی با کارآیی بالا مانند ;کنسول بازی(video – game) و جعبه اتصال کامپیوتر به تلویزیون کابلی (set – top boxes) به بازار عرضه شدند.

این پیشرفت‌های مرحله به مرحله باعث تغییر نگاه ها به کامپیوتر، کاربردهای آن و بازار مصرف آن در آغاز هزارۀ جدید می‌شود.کمتر از ۲۰ سال از ظهور کامپیوترهای شخصی می‌گذرد و شاهد تغییرات زیادی در کاربردهای آنها هستیم. این تغییرات باعث شده که امروزه سه بازار عمده برای کامپیوتر داشته باشیم که هر کدام کاربردها، نیازمندیها و فناوری خاص خود را دارد.

## کامپیوترهای رومیزی (Desktop Computing)

اولین و بزرگترین بازار از نظر هزینه‌ای که صرف می‌شود، کامپیوترهای رومیزی می‌باشد. کامپیوترهای رومیزی از سیستم‌های ارزان قیمت که کمتر از هزار دلار قیمت دارند تا سیستم‌های کاری (workstation)های با سازماندهی قوی که بیش از ده هزار دلار قیمت دارند می‌باشند. در این محدود از نظر قیمت و کارآیی، نسبت کارآیی به قیمت(price –performance) بهینه مورد نظر می‌باشد. این ترکیب کارآیی (کارآیی با قدرت محاسباتی و قدرت گرافیکی سنجیده می‌شود) و قیمت باعث شده که مصرف کنندگان و طراحان کامپیوتر به سمت این کامپیوترها جذب شوند. در نتیجه سیستم‌های رومیزی جایی هستند که جدیدترین و کارآترین ریزپردازنده‌ها در آنها ظاهر شده و معمولاً اولین جایی هستند که ریزپردازنده‌ها و سیستم‌های ارزان شده ظاهر می‌شوند. (در قسمت ۱-۴ دربارۀ عوامل تأثیرگذار بر قیمت کامپیوتر بحث می‌شود.)

کامپیوترهای رومیزی بخوبی بر اساس کاربرد و کارآیی بررسی می‌شوند چون ابزارهای بررسی کارآیی زیادی برای آنها روی شبکه اینترنت وجود دارد. در بخش ۱-۹ بحث می‌شود که امروزه در دنیای کامپیوترهای شخصی فقط به کلاک سیستم توجه می‌شود که این مسأله باعث گمراهی مصرف‌کننده می‌شود.

## سرویس دهنده‌ها (Servers)

با رواج کامپیوترهای رومیزی نقش سرویس دهنده‌ها که بطور مطمئنی فایل‌ها و سرویس‌ها را ارائه دهند، بیشتر شد. وجود شبکه جهانی اینترنت به این رشد شتاب بیشتری بخشید. این سرورها که ستون فقرات (backbone) شبکه هستند، جایگزین کامپیوترهای بزرگ (Mainframe) شده‌اند.

برای سرورها مشخصات دیگری مهم هستند. اولین مشخصه، "در دسترس بودن" (Availability) است، "در دسترس بودن" یعنی اینکه سیستم بطور مطمئن و مؤثری ارائه سرویس کند. این عبارت با "قابلیت اطمینان" (Reliability) که می‌گوید سیستم هیچگاه خراب نمی‌شود، تفاوت دارد. اجزای یک سیستم بزرگ خواه ناخواه خراب می‌شوند ولی هدف در یک سرور در دسترس بودن سیستم حتی در صورت خرابی بعضی از اجزا می‌باشد که این کار با افزونگی (Redundancy) انجام می‌شود .

چرا در دسترس بودن حیاتی(Crucial) می‌باشد؟ سرورهایی که یاهو را سرویس می‌دهند یا سفارشات کمپانی Cisco را می‌پذیرند و یا حراج ebay را انجام می‌دهند را در نظر بگیرید. این سیستم‌ها باید هفت روز هفته بطور۲۴ ساعته (Seven days a week, 24 hours a day) کار کنند.

خراب شدن چنین سرویس دهنده‌هایی با خراب شدن یک کامپیوتر رومیزی تکی تفاوت دارد. گرچه تخمین هزینهٔ خرابی این سرویس دهنده‌ها مشکل می‌باشد ولی شکل دو تخمینی از هزینه خرابی ارائه می‌دهد و فرض می‌کند که خرابی در زمان عادی رخ دهد. همانطور که دیده می‌شود هزینه در

دسترسی نبودن خیلی بالا می‌باشد. در شکل دو فقط خسارات مادی در نظر گرفته شده و عدم رضایت مشتریان منظور نشده است!

| Application | Cost of downtime per hour (thousands of $) | Annual losses (millions of $) with downtime of 1% (87.6 hrs/yr) | 0.5% (43.8 hrs/yr) | 0.1% (8.8 hrs/yr) |
|---|---|---|---|---|
| Brokerage operations | $6450 | $565 | $283 | $56.5 |
| Credit card authorization | $2600 | $228 | $114 | $22.8 |
| Package shipping services | $150 | $13 | $6.6 | $1.3 |
| Home shopping channel | $113 | $9.9 | $4.9 | $1.0 |
| Catalog sales center | $90 | $7.9 | $3.9 | $0.8 |
| Airline reservation center | $89 | $7.9 | $3.9 | $0.8 |
| Cellular service activation | $41 | $3.6 | $1.8 | $0.4 |
| Online network fees | $25 | $2.2 | $1.1 | $0.2 |
| ATM service fees | $14 | $1.2 | $0.6 | $0.1 |

شکل ۲

هزینه در دسترس نبودن سیستم‌ها

دومین ویژگی کلیدی سرویس دهنده‌ها، مقیاس پذیری (Scalability) آنها می‌باشد. سیستم‌های سرویس‌دهنده معمولاً در طول حیات خود باید گسترش یابند تا بتوانند به عملیات مورد نیاز پاسخ دهند. بنابراین یک سرویس دهنده باید امکانات گسترش حافظه، پهنای باند ورودی –خروجی و قدرت محاسباتی را داشته باشد.

در نهایت سرویس دهنده‌ها باید کارآیی (throughput) بالایی داشته باشند. تعداد عملیات در دقیقه یا تعداد صفحات وب در ثانیه، معیار بررسی کارآیی می‌باشد. پاسخ دهی به درخواست‌های جداگانه مهم است ولی برای‌اندازه‌گیری کارآیی تعداد صفحات در واحد زمان بررسی می‌شود.

## کامپیوترهای همراه یا جاسازی شده (Embedded Computers)

کامپیوترهای جاسازی شده، کامپیوترهایی هستند که در وسایل دیگر قراردارند و در نگاه اول وجود کامپیوتر آشکار نیست. بازار این نوع کامپیوترها سریعترین رشد را در دنیای کامپیوتر دارد.

این وسایل از ماشین‌های روزمره شامل اجاق‌های ماکروویو، ماشین‌های لباسشویی، چاپگرها، سؤیچ‌های شبکه و اکثر اتومبیل‌ها که ریزپردازنده‌های داخلی دارند ، تا وسایل دستی دیجیتال مانند کارت‌های هوشمند، تلفن همراه و کامپیوترهای جیبی (Palmtop) تا کنسول‌های بازی (Videogame) و Set-top boxes هستند. گرچه در بعضی از کاربردها مانند کامپیوترهای جیبی، کامپیوترها توسط کاربر قابل برنامه ریزی هستند، ولی در اکثر کاربردها برنامه ریزی فقط در هنگام ساخت اولیه انجام می‌شود. بنابراین کاربرد باید متناسب با پردازنده و سیستم باشد. معمولاً در این کاربردها برای قسمت‌های کلیدی برنامه‌ها از زبان اسمبلی استفاده می‌شود. گرچه فشار بازار و روشهای خوب مهندسی نرم افراز استفاده از زبان اسمبلی را محدود می کند.

استفاده از زبان اسمبلی، وجود یک سیستم عامل استاندارد و وجود کدهای پایهٔ زیاد باعث شده است که نیاز به کدهای همساز(instruction set compatibility) یک مسالهٔ مهم در کامپیوترهای جاسازی شده باشد. مانند اکثر کاربردهای دیگر هزینه نرم افزار قسمت عمدهٔ هزینه‌ها را در کامپیوترهای جاسازی شده شامل می‌شود.

کامپیوترهای جاسازی شده محدوده وسیعی دارند از پردازنده‌های ۸ و ۱۶ بیتی که کمتر از یک دلار قیمت دارند تا پردازنده‌های ۳۲ بیتی با امکان پردازش ۵۰ میلیون دستور العمل در ثانیه که کمتر از ده دلار قیمت دارند تا پردازنده‌های اختصاصی که صدها دلار قیمت داشته و می‌توانند میلیاردها دستور را در ثانیه اجرا کنند و در سوئیچ‌های شبکه و کنسول‌های بازی استفاده می‌شوند. معمولاً در کامپیوتر جاسازی شده، قیمت خیلی مهم می‌باشد. معمولاً هدف اصلی، برآورد کردن کارآیی با حداقل قیمت بوده و کارآرایی بالاتر با قیمت بیشتر مورد نظر نیست. معمولاً در سیستم‌های جاسازی شده کارآیی مورد نیاز

یک کارآیی بلادرنگ(real-time) می‌باشد. در کاربردهای بلادرنگ باید قسمتی از عملیات در زمان مشخصی انجام شود. مثلاً در set – top box زمان پردازش هر فریم از تصویر محدود می‌باشد چون پردازنده باید در زمان کوتاهی فریم بعدی را پذیرفته و پردازش کند. در بعضی از کاربردها متوسط زمان پاسخ دهی مورد نظر می‌باشد و در بعضی از مواقع می‌تواند از حدی بیشتر باشد. به این مسأله Soft real – time می‌گویند.

کار آیی سیستم‌های بلا درنگ وابسته به کاربرد می‌باشد. می‌توان با ارزیابی از طریق کاربرد یا برنامه‌های استاندارد ارزیابی(benchmark) کارآیی را ارزیابی کرد. برنامه هایی برای ارزیابی برنامه‌های جاسازی شده از برنامه‌های کوچک تا برنامه‌های هزاران خطی وجود دارد.

دو ویژگی دیگر کاربردهای جاسازی شده عبارتند از نیاز به بهینه کردن حافظه و بهینه کردن توان مصرفی، در بعضی از سیستم‌های جاسازی شده حافظه بخش مهمی‌از هزینه سیستم را تشکیل می‌دهد و ضروری است که‌اندازه حافظه کاهش یابد. در بعضی از کاربردها باید برنامه‌ها در حافظه داخلی پردازنده قرار گیرند یا اینکه برنامه در یک تراشه کوچک حافظه قرار گیرد. در هر کدام از این موارد باید اندازهٔ کد کوچک شود چون اندازه داده توسط کاربرد از قبل مشخص شده است. بعضی از معماریهای کامپیوتر دارای دستور العمل هایی برای کاهش اندازهٔ کد هستند. حافظهٔ بیشتر به معنی توان مصرفی بیشتر نیز بوده و توان مصرفی از پارامترهای بحرانی در سیستم‌های جاسازی شده است. گرچه فقط در سیستم‌های کوچک از باتری استفاده می‌شود ولی مصرف توان کمتر باعث می‌شود بتوان از بسته بندی پلاستیکی به جای سرامیکی در ساخت پردازنده استفاده کرده و نیازی به استفاده از فن برای پردازنده نباشد.

یکی از مسائل دیگر در سیستم‌های جاسازی شده استفاده از پردازنده همراه با مدارات مورد نیاز کاربرد می‌باشد. همچنین باید نرم افزارهای مورد نیاز برای کاربرد نوشته شود. مسائل سیستم‌های جاسازی شده به سه روش زیر حل می‌شود:

۱ – طراح از ترکیب نرم افزار و سخت افزار استفاده می‌کند یعنی با استفاده از سخت افزارهای اختصاصی و اضافه کردن پردازنده به آن، مدار مورد نیاز را طراحی می‌کند.

۲ – استفاده از پردازنده‌های موجود و نوشتن نرم افراز مورد نیاز

۳ – استفاده از پردازنده‌های پردازش سیگنال دیجیتال (Digital Signal Processor) (DSP) که اختصاصاً برای کاربردهای پردازش سیگنال طراحی شده‌اند.

هدف این کتاب طراحی و استفاده از پردازنده‌های معمولی بوده و استفاده از سخت افزارهای خاص یا DSP از اهداف این کتاب نیست.

| Feature | Desktop | Server | Embedded |
|---|---|---|---|
| Price of system | $1000–$10,000 | $10,000–$10,000,000 | $10–$100,000 (including network routers at the high end) |
| Price of microprocessor module | $100–$1000 | $200–$2000 (per processor) | $0.20–$200 (per processor) |
| Microprocessors sold per year (estimates for 2000) | 150,000,000 | 4,000,000 | 300,000,000 (32-bit and 64-bit processors only) |
| Critical system design issues | Price-performance, graphics performance | Throughput, availability, scalability | Price, power consumption, application-specific performance |

شکل ۳ –خلاصهٔ سه کلاس مختلف کامپیوترها و مشخصات آن سیستم ها

به محدود وسیع قیمت در سیستم‌های سرویس دهنده و سیستم‌های جاسازی شده توجه کنید. در مورد سرویس دهنده‌ها این محدوده بعلت نیاز به چند پردازنده‌های پردازش نقل و انتقالات یا کاربردهای صفحات وب می‌باشد. در سیستم‌های جاسازی شده یکی از کاربردهای مرز دانش، مسیر یاب‌های شبکه می‌باشد که نیاز به چندین پردازنده و مقدار زیادی حافظه دارد. تعداد پردازنده‌های جاسازی شده مصرفی در سال ۲۰۰۰ اگر پردازنده‌های ۸ و ۱۶ بیتی را در نظر بگیریم به یک میلیارد پردازنده می‌رسد. در حقیقت بیشترین پردازنده‌ای که فروخته می‌شود پردازنده‌های ۸ بیتی است که توسط اینتل فروخته می‌شود. سرورهای کوچک آنهایی که کمتر از ۵۰۰۰ دلار قیمت دارند و کامپیوترهای رومیزی تقریباً یکسان هستند.

## وظیفه طراحان کامپیوتر

وظیفه یک طرح کامپیوتر خیلی پیچیده می‌باشد، باید مشخص شود که چه ویژگیهایی برای یک ماشین جدید مهم می‌باشد، سپس ماشینی را طراحی کند که کارآیی حداکثری داشته ولی قیمت و توان مصرفی آن معقول باشد. این کار جنبه‌های مختلفی دارد از طراحی مجموعه دستورالعمل‌ها، سازماندهی عملیات، طراحی مدارات منطقی، تا پیاده سازی نهایی، که پیاده سازی شامل طراحی مدارات مجتمع، بسته بندی، تغذیه و خنک سازی می‌باشد.

بهینه کردن طراحی نیاز به آشنایی با محدودهٔ وسیعی از فن آوریهای مختلف از کامپایلر و سیستم عامل تا طراحی مدارات منطقی و بسته‌بندی دارد.

در گذشته معماری کامپیوتر فقط محدود به طراحی مجموعه دستورالعمل‌ها بود و بقیه مسائل طراحی کامپیوتر را پیاده سازی می‌نامیدند و پیاده سازی را خیلی مهم نمی‌گرفتند. ولی این طرز فکر نه تنها اشتباه می‌باشد بلکه سبب خطاهای طراحی در مجموعه‌های جدید دستورالعمل‌ها می‌شود. کار معمار یا طراح کامپیوتر در رابطه با بقیه کارها همان قدر مهم است که طراحی مجموعه دستورالعمل‌ها مهم می‌باشد. این مسئله امروز حادتر شده است چون اختلاف مجموعه دستورالعمل‌ها کاهش یافته و سه محدوهٔ کاربردی متفاوت وجود دارد.

در این کتاب منظور از معماری مجموعه دستورالعمل ها اشاره به مجموعهٔ دستوراتی است که توسط برنامه نویس قابل دسترسی مستقیم می‌باشد. مجموعه دستور العمل‌ها ارتباط بین سخت افزار و نرم افزار هستند. پیاده سازی یک ماشین شامل دو قسمت سازماندهی(Organization) و سخت‌افزار(Hardware)می‌باشد.

عبارت سازماندهی(Organization) شامل نگاه طراح کامپیوتر از بالا می‌باشد و شامل سیستم حافظه، ساختار گذرگاه مشترک، و طراحی پردازنده مرکزی داخلی CPU (که عملیات ریاضی، منطقی، پرش و انتقال داده را پیاده سازی می‌کند) می‌باشد. مثلاً دو سیستم جاسازی شده می‌توانند مجموعه دستورالعمل‌های یکسان داشته ولی سازماندهی متفاوتی داشته باشند، مانند دو سیستم NEC VR 4122 و NEC VR 5432 که هر دو پردازنده، مجموعه دستور العمل‌های MIPS64 را پیاده سازی می‌کنند ولی سازمان خط لوله و حافظه نهان متفاوتی دارند. بعلاوه ۴۱۲۲ دستورالعمل‌های ممیز شناور را به جای سخت‌افزار با نرم افزار پیاده سازی می‌کند. منظور از سخت‌افزار یک کامپیوتر، اشاره به موجودیت آن می‌باشد که شامل طراحی مدارات منطقی و فناوری بسته بندی آن می‌باشد. معمولاً یک سری از ماشین‌ها دارای معماری دستور العمل‌های یکسانی بوده و تقریباً سازماندهی مشابهی دارند ولی از نظر پیاده سازی سخت‌افزاری متفاوت می‌باشند. مثلاً پنتیوم ۲ و سلرون تقریباً یکسان هستند ولی کلاک ساعت و سیستم حافظه متفاوتی دارند و این باعث شده که سلرون برای ساخت کامپیوترهای ارزان قیمت مناسب باشد. در این کتاب منظور از معماری کامپیوتر هر سه جنبهٔ آن یعنی طراحی مجموعه دستورالعمل‌ها، سازماندهی و سخت‌افزار می‌باشد.

یک طراح کامپیوتر باید کامپیوتری طراحی کند که علاوه بر تأمین عملیات مورد نیاز، اهداف مربوط به کارآیی، تغذیه و قیمت را نیز تأمین کند. معمولاً لازم است که مشخص شود عملیات مورد نیاز چیست. نیازمندیها، ویژگی خاصی است که توسط بازار تعیین می‌گردد. نرم افزارهای کاربردی با مشخص کردن اینکه کامپیوتر چگونه استفاده شود، بعضی از این عملیات مورد نیاز را مشخص می‌کنند. اگر محدودهٔ وسیعی از نرم افزارها برای معماری مجموعه دستورات خاصی وجود دارد، طراح تصمیم

می‌گیرد که این مجموعه از دستورات را در کامپیوتر جدید پیاده سازی کند. وجود درصد بزرگی از بازار برای کاربردی خاص سبب می‌شود که طراح خود را با این بازار وفق دهد. شکل ۴ خلاصه بعضی از نیازهایی را که باید درطراحی جدید در نظر گرفته شود، نشان می‌دهد.

| Functional requirements | Typical features required or supported |
|---|---|
| *Application area* | *Target of computer* |
| General-purpose desktop | Balanced performance for a range of tasks, including interactive performance for graphics, video, and audio |
| Scientific desktops and servers | High-performance floating point and graphics |
| Commercial servers | Support for databases and transaction processing; enhancements for reliability and availability; support for scalability |
| Embedded computing | Often requires special support for graphics or video (or other application-specific extension); power limitations and power control may be required |
| *Level of software compatibility* | *Determines amount of existing software for machine* |
| At programming language | Most flexible for designer; need new compiler |
| Object code or binary compatible | Instruction set architecture is completely defined—little flexibility—but no investment needed in software or porting programs |
| *Operating system requirements* | *Necessary features to support chosen OS* |
| Size of address space | Very important feature; may limit applications |
| Memory management | Required for modern OS; may be paged or segmented |
| Protection | Different OS and application needs: page vs. segment protection |
| *Standards* | *Certain standards may be required by marketplace* |
| Floating point | Format and arithmetic: IEEE 754 standard, special arithmetic for graphics or signal processing |
| I/O bus | For I/O devices: Ultra ATA, Ultra SCSI, PCI |
| Operating systems | UNIX, PalmOS, Windows, Windows NT, Windows CE, CISCO IOS |
| Networks | Support required for different networks: Ethernet, Infiniband |
| Programming languages | Languages (ANSI C, C++, Java, FORTRAN) affect instruction set |

شکل ۴ –خلاصهٔ بعضی از نیازمندیهای عملیاتی که بر معماری کامپیوتر تاثیر دارند.

ستون سمت چپ، مشخص کننده نیازمندی مورد نیاز بوده و در ستون سمت راست، مثالی از ویژگی مورد نیاز ذکر شده است.

بعد از اینکه مجموعه نیازمندیهای عملیاتی مورد نیاز مشخص شد، معمار کامپیوتر سعی در بهینه کردن طرح می‌کند. اینکه چه مسائلی باید بهینه شود بستگی به انتخاب روش‌اندازه گیری مناسب دارد. در مورد کامپیوترهای رومیزی بهینه شدن نسبت هزینه به کارآیی مورد نظر می‌باشد، در مورد

سرویس‌دهنده‌ها در دسترس بودن، مقیاس پذیری، کارآیی و قیمت و در مورد کامپیوترهای جاسازی شده قیمت و تغذیه مورد نظر می‌باشد.

این تفاوت‌ها و تنوع و اندازه متفاوت بازار باعث کوشش‌های کاملاً متفاوت طراحی شده است. در مورد کامپیوترهای رومیزی اکثر تلاشها بر طراحی یک ریزپردازنده پیشرفته و سیستم گرافیکی و ورودی و خروجی متناسب با آن می‌باشد. در سرویس دهنده ها توجه به ریزپردازنده‌های پیشرفته و اکثراً به صورت چند پردازنده و مقیاس پذیر و با سیستم‌های ورودی – خروجی با قابلیت دسترس پذیری بالا می‌باشد. در نهایت، در سیستم‌های جاسازی شده امروزی، هدف استفاده از ریزپردازنده‌های مدرن برای داشتن حداکثر کارآیی با حداقل قیمت و توجه به تغذیه و در بعضی ار کاربردها توجه به پردازش ویدئو یا گرافیک سطح بالا می‌باشد.

علاوه بر قیمت و کارآیی طراحان باید فناوری مورد استفاده در پیاده سازی و استفاده از کامپیوتر را در نظر داشته باشند، در نظر گرفتن این نکات علاوه بر کاهش هزینه‌های بعدی، باعث عمر مفید بیشتری برای طرح می‌شود. در دو بخش بعدی این مسائل از دیدگاه فناوری و دیدگاه قیمت بررسی می‌شوند.

## ۱ – ۳ دیدگاه فناوری

برای اینکه یک معماری مجموعه دستور العمل ها موفق باشد، باید طراحی بتواند با پیشرفت‌های سریع فناوری کامپیوتر تطبیق یابد. بعلاوه، یک مجموعهٔ دستورات موفق باید ده‌ها سال دوام داشته باشد، هستهٔ اصلی کامپیوترهای بزرگ IBM هنوز بعد از ۳۵ سال استفاده می‌شود. یک معمار کامپیوتر باید تغییرات فناوری را مد نظر قرار دهد تا عمر مفید یک کامپیوتر موفق بیشتر شود.

برای برنامه ریزی تکامل کامپیوتر، طراح باید بطور خاص پیشرفت در فناوری پیاده سازی را در نظر داشته باشد. چهار فناوری پیاده سازی، که بطور سریعی تغییر کرده‌اند و توجه به آنها در پیاده سازی‌های مدرن حیاتی است عبارتند از:

### فناوری مدارات منطقی مجتمع (IC)

تجمع ترانزیستورها تقریباً هر سال ۳۵٪ افزایش می‌یابد یعنی هر ۴ سال ۴ برابر می‌شود. افزایش دراندازهٔ ویفرها آهسته‌تر بوده و ۱۰ تا ۲۰ درصد در سال می‌باشد. ترکیب این دو مسأله باعث رشد تعداد ترانزیستورها در یک تراشه با حدود ۵۵٪ در سال شده است. ولی سرعت اجزا به میزان کمتری افزایش یافته است.

### حافظه‌های پویای نیمه هادی (DRAM)

فشردگی حافظه‌های پویا ۴۰ تا ۶۰ درصد در سال افزایش می‌یابد یعنی هر ۳ تا ۴ سال چهار برابر می‌شود. زمان چرخش (cycle time) پیشرفت کندتری دارد. تقریباً در ۱۰ سال یک سوم شده است. پهنای باند در مقایسه با کاهش تأخیر برای هر تراشه تقریباً دو برابر شده است. همچنین تغییرات در واسط DRAM باعث بهبود باند شده است .

### فناوری دیسک‌های مغناطیسی

امروزه فشردگی دیسک تقریباً هر سال ۱۰۰٪ افزایش می‌یابد و در دو سال چهار برابر می‌شود. ولی قبل از ۱۹۹۰ میزان افزایش فشردگی ۳۰٪ در سال بود یعنی در ۳ سال حجم دیسک دو برابر می‌شد. انتظار می‌رود که در سالهای آینده نیز فشردگی دیسک افزایش یابد. زمان دسترسی (Access time) نیز در طی ۱۰ سال تقریباً یک سوم شده است.

### فناوری شبکه

کارآیی یک شبکه وابسته به کارآیی سوئیچ ها و سیستم انتقال داده می‌باشد. هر دومسأله زمان پاسخ و پهنای باند را می‌توان گسترش داد ولی امروزه پهنای باند مهم‌تر شده است. طی سالهای متمادی رشد شبکه کند بود مثلاً ۱۰ سال طول کشید تا فناوری Ethernet از ۱۰Mb به ۱۰۰MB برسد.افزایش اهمیت شبکه باعث رشد سریعتر شد و باعث شد که ۵ سال بعد پهنای باند Ethernet از ۱۰۰mb به ۱Gb برسد. با استفاده از فیبرهای نوری و بهبود سخت افراز سوئیچ‌ها رشد اینترنت در ایالات متحده سریعتر بوده است (تقریباً هر سال پهنای باند آن دو برابر شده است).

با توجه به گسترش سریع فناوری باید ریزپردازنده‌ها طوری طراحی شوند که عمر مفید ۵ سال یا بیشتر داشته باشند. حتی در طول زمانی تولید یک سیستم کامپیوتری (۲ سال برای طراحی و ۲ تا ۳ سال برای ساخت) فناوری‌های کلیدی مانند DRAM تغییرات زیادی دارند و طراحان باید طراحی خود را برای فناوری‌های آینده انجام دهند تا مطمئن باشند که در زمانی که محصول آنها به بازار عرضه می‌شود آن فناوری به صرفه می‌باشد. معمولاً قیمت به همان نسبت افزایش فشردگی، کاهش می‌یابد.

ولی پیشرفت فناوری پیوسته نیست و اثر این پیشرفت در زمانهای گسسته دیده می‌شود، یعنی تا زمانی که به آستانه‌ای برسد که اجازهٔ یک محصول جدید را بدهد. مثلاً فناوری MOS در اوایل دههٔ ۱۹۸۰ می‌توانست ۲۵۰۰۰ تا ۵۰۰۰۰ ترانزیستور را در یک IC جا دهد، بنابراین امکان ساخت یک پردازنده ۳۲ بیتی روی یک تراشه بوجود آمد. در اواخر دههٔ ۱۹۸۰ امکان قرار دادن حافظه نهان سطح یک در داخل تراشه فراهم شد.

با نادیده گرفتن ارتباطات داخلی پردازنده و بین پردازنده و حافظه نهان، افزایش زیادی در کارآیی ـ هزینه و کارآیی- توان ایجاد شد. این طراحی تا زمانی که فناوری به حد مشخصی نرسیده بود، امکان

پذیر نبود. رسیدن فناوری به این آستانه‌ها امری بعید نبوده و تأثیر مهمی‌روی تصمیم گیری برای طراحی دارد.

## اثر مقیاس پذیری بر کارآیی ترانزیستور، اتصالات و توان در مدارات مجتمع

فرآیند مدارات مجتمع با ویژگی‌اندازه (size) مشخص می‌شود، که این ویژگی حداقل اندازهٔ یک ترانزیستور یا اتصال در جهت x یا y می‌باشد. ویژگی‌اندازه از ۱۰ میکرون در سال ۱۹۸۱ به ۰/۱۸ میکرون در سال ۲۰۰۱ رسیده است. چون تعداد ترانزیستورها در یک میلی متر مربع از نیمه هادی سیلیکون شمرده شده و وابسته به مساحت یک ترانزیستور است، بنابراین وقتی که اندازهٔ ترانزیستور خطی کاهش یابد، چگالی ترانزیستورها تصادی زیاد می‌شود. ولی افزایش کارآرایی ترانزیستور پیچیده‌تر می‌باشد. وقتی که ویژگی‌اندازه کاهش می‌یابد، ابعاد وسیله بصورت تصاعدی در جهت افقی و عمودی و همچنین در جهت عمقی کاهش می‌یابد. کاهش بُعد عمقی نیاز به کاهش ولتاژ عملیاتی دارد تا کارکرد درست و قابلیت اطمینان ترانزیستور تضمین شود. این ترکیب مقیاس باعث یک ارتباط پیچیده بین کارآرایی ترانزیستور و ویژگی اندازه می‌شود. بعنوان یک تقریب اولیه کارآیی ترانزیستور بطور خطی با کاهش ویژگی‌اندازه، افزایش می‌یابد.

این حقیقت که تعداد ترانزیستورها تصاعدی افزایش یافته و همچنین کارآیی آنها خطی افزایش می‌یابد دقیقاً همان مواردی است که طراحان کامپیوتر به هر دوی آنها نیاز دارند.

در ابتدای پیدایش ریزپردازنده‌ها، افزایش چگالی ترانزیستورها باعث حرکت سریع از پردازنده‌های ۴ بیتی به ۸ بیتی، سپس به ۱۶ بیتی و آنگاه ۳۲ بیتی بود. اخیراً بهبود چگالی باعث معرفی پردازنده‌های ۶۴ بیتی شده و همچنین باعث رشد در زمینه‌های خط لوله و حافظه نهان شده است.

گرچه کارآیی ترانزیستورها با کاهش ویژگی اندازه، افزایش می‌یابد، ولی اتصالات داخلی مدارات مجتمع این طور نیستند. در عمل تأخیر یک سیگنال در یک سیم متناسب با مقاومت و خازن آن سیم می‌باشد. با کاهش ویژگی‌اندازه، طول اتصالات کوچکتر شده ولی مقاومت آنها و خازن آنها در طول واحد، بدتر می‌شود. این ارتباط پیچیده می‌باشد یعنی وابسته به ابعاد هندسی سیم، باری که روی آن سیم وجود دارد و همچنین وابسته به همسایه‌های آن سیم می‌باشد. بعضی از پیشرفت‌های ناگهانی در فرآیند تولید مانند معرفی مس، باعث بهبود تأخیر اتصالات شدند. بطور کلی، تأخیر اتصالات در اثر مقیاس پذیری، در مقایسه با کارآیی ترانزیستورها بدتر می‌شود. این مسأله چالش‌های بیشتری را برای طراحان فراهم می‌کند. در چند سال اخیر، تأخیر اتصالات یکی از محدودیت‌های اصلی برای مدارات مجتمع بزرگ و معمولاً بحرانی‌تر از تأخیر سوئیچ ترانزیستورها بود. قسمت بیشتر و بیشتری از سیکل کلاک توسط تأخیر سیگنال در اتصالات تلف می‌شود. در سال ۲۰۰۱، پردازنده پنتیوم ۴ روش جدیدی بکار بست و ۲ مرحله از بیش از ۲۰ مرحله خط لوله‌ی خود را برای انتقال سیگنالها در طول تراشه مصرف کرد.

با کوچکتر شدن اجزا، تغذیه نیز یکی از چالشها می‌باشد. برای ریزپردازنده‌های مدرن CMOS، عمدهٔ انرژی مصرفی در ترانزیستورهای سؤیچ می‌باشد. انرژی مصرفی در هر ترانزیستور متناسب با حاصل ضرب خازن بار ترانزیستور، فرکانس سؤیچ شدن و مربع ولتاژ می‌باشد. با حرکت از یک فرآیند به فرآیند پیشرفته‌تر، تعداد ترانزیستورهای سؤیچ و فرکانسی که آنها سؤیچ می‌کنند افزایش یافته ولی خازن بار و ولتاژ کاهش می‌یابد اما در نهایت توان مصرفی افزایش می‌یابد. ریزپردازنده‌های اولیه یک دهم وات توان مصرف می‌کردند، در صورتی که یک پنتیوم ۴ با فرکانس ۲ گیگا هرتز، ۱۰۰ وات توان مصرف می‌کند. ریزپردازنده‌های مربوط به ایستگاه‌های کاری سریع و سرویس دهنده‌ها حدود ۱۰۰ تا ۱۵۰ وات توان

مصرف می‌کنند. توزیع توان، خنک کردن تراشه و جلوگیری از آسیب رسیدن به پردازنده در اثر حرارت از چالش‌های بزرگ می‌باشد و به نظر می‌رسد که در آینده توان مصرفی بیش از تعداد ترانزیستورها باعث محدود شدن طراحی‌ها می‌شود.

## ۱ – ۴ هزینه، قیمت و رابطه بین آنها

گرچه در اکثر طراحی‌های کامپیوتر- بخصوص سوپر کامپیوترها- هزینه اهمیت کمی‌دارد ولی طراحی‌هایی که در آنها قیمت نقش اصلی را دارد بطور چشمگیری افزایش یافته‌اند. بیش از نیمی از کامپیوترهای شخصی فروخته شده در سال ۱۹۹۹ کمتر از هزار دلار قیمت داشته‌اند و متوسط قیمت یک ریزپردازنده ۳۲ بیتی برای کاربردهای جاسازی شده در حدود ده دلار می‌باشد. بعلاوه در پانزده سال اخیر، پیشرفت فناوری باعث کاهش قیمت ها در کنار افزایش کارآیی در صنعت کامپیوتر شده است.

کتاب‌های درسی معمولاً این نکته که قیمت به تنهایی نیمی از قیمت – کارآیی می‌باشد را نادیده می‌گیرند. قیمت ها تغییر می‌کنند و این مسئله در جاهای مختلف صنعت وجود دارد ولی هنوز فهم قیمت و نقش اساسی آن برای یک طراح کامپیوتر مهم می‌باشد تا بتواند تصمیم‌های هوشمندانه‌ای اتخاذ کرده و در مورد استفاده یا عدم استفاده از یک ویژگی جدید در جاهایی که قیمت مهم است تصمیم بگیرد. (فرض کنید که یک معمار بخواهد آسمان خراش طراحی کند ولی هیچ اطلاعاتی از قیمت تیرآهن و سیمان نداشته باشد!)

این بخش روی هزینه‌ها و قیمت، بخصوص روی ارتباط بین قیمت و هزینه تمرکز دارد. قیمت یعنی نرخی که کالای تمام شده به آن نرخ فروخته می‌شود ولی هزینه، بهایی است که برای تولید آن کالا پرداخت شده است که این بها شامل هزینه‌های بالاسری نیز می‌باشد. همچنین عوامل اصلی که روی هزینه‌ها اثر داشته و تغییر آنها در طول زمان شرح داده می‌شود. گرچه این هزینه‌ها در طول زمان تغییر

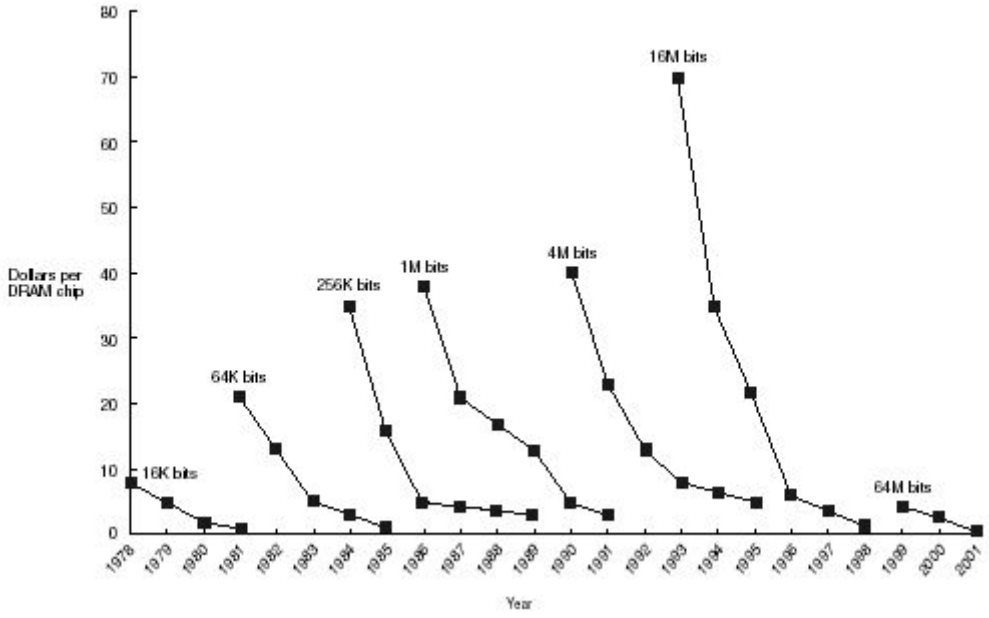
می‌کنند، ولی اصول مربوط به هزینه در طول زمان ثابت است. این بخش شما را با این مسائل از طریق شرح بعضی از عوامل مهم که هزینه‌های طراحی یک کامپیوتر را تحت تأثیر قرار می‌دهند و نحوهٔ تغییر آنها در طول زمان، آشنا می‌کند.

## زمان، حجم تولید و تجاری‌شدن

هزینه تولید اجزای کامپیوتر به مرور زمان کاهش می‌یابد حتی اگر فناوری پیاده سازی آن تغییر زیادی نداشته باشد. نکته اصلی که باعث کاهش هزینه‌ها می‌شود منحنی یادگیری است که باعث می‌شود هزینهٔ تولید به مرور زمان کاهش یابد. منحنی یادگیری خودش با تغییر در بهبود کالا یعنی درصد اجزایی که از تست نهایی موفق خارج می‌شوند، اندازه گیری می‌شود. این جزء می‌تواند یک تراشه، یک برد یا یک سیستم باشد. طرحی که میزان موفقیت آن دو برابر باشد باعث نصف شدن هزینه‌ها می‌شود.

فهم اینکه منحنی یادگیری چگونه باعث بهبود موفقیت می‌شود کلیدی برای داشتن تصوری از هزینه‌ها در طول حیات یک محصول می‌باشد. مثالی برای منحنی یادگیری در عمل، قیمت به ازای یک میلیون بایت برای حافظه‌های پویا(DRAM) می‌باشد که ۴۰٪ در سال کاهش می‌یافت. چون قیمت DRAM نسبت نزدیکی با هزینه تولید آن دارد، به جز زمانی که یک رکود رخ می‌دهد، هزینه و قیمت DRAM خیلی به هم نزدیک می‌باشد. در حقیقت، در بعضی از دوره‌های زمانی(مثلاً در ابتدای سال ۲۰۰۱) قیمت حتی از هزینه تولید کمتر بود، البته تولید کنندگان مایلند که این زمانها به ندرت اتفاق افتاده و کوتاه باشند. شکل ۵ قیمت یک تراشه DRAM را در طول عمر آن رسم کرده است. از زمان شروع یک پروژه تا عرضهٔ آن به بازار که معمولاً دو سال می‌باشد، هزینه یک DRAM جدید با فاکتوری بین ۵ تا ۱۰ با نرخ ثابت دلار سقوط می‌کند. چون همهٔ اجزا با نرخ یکسان تغییر نمی‌کنند، طراحی با توجه به

پیش بینی هزینه‌ها نتیجهٔ متفاوتی از نظر قیمت – هزینه در مقایسه با هزینه‌های جاری دارد. توضیح شکل ۵ برخی از مسائل دخیل در تجارت و قیمت DRAM را بیان کرده است

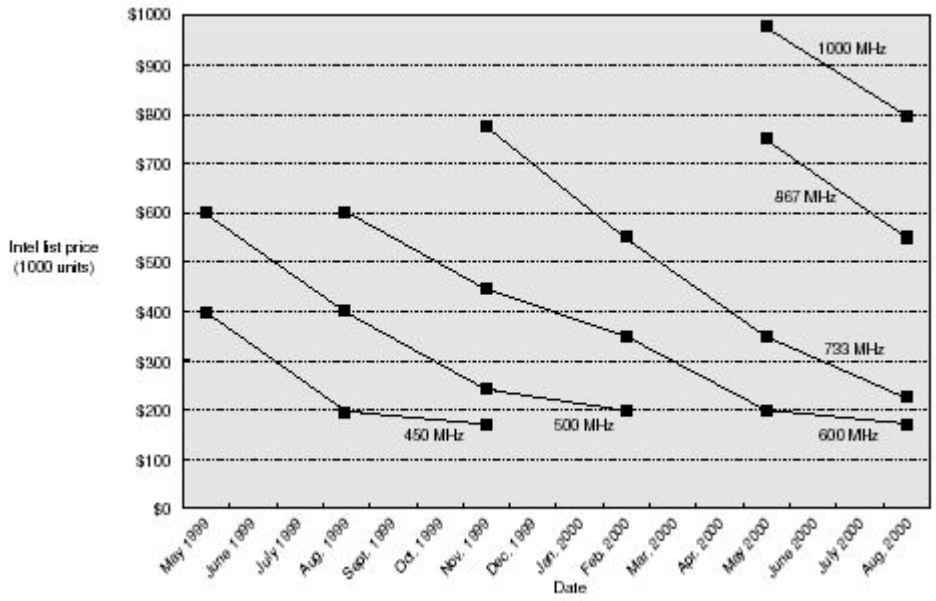



شکل ۵ - قیمت ۶ نسل از DRAM (از 16 kبیت تا ۶۴مگابیت)

این نمودار نشان دهندهٔ منحنی یادگیری در طول زمان بر حسب قیمت دلار در سال ۱۹۷۷ می‌باشد. یک دلار ۱۹۷۷ حدود ۲/۹۵ دلار در سال ۲۰۰۱ ارزش داشت. هزینهٔ یک مگابایت حافظه بطور باورنکردنی در این سالها از ۵ هزار دلار در سال ۱۹۷۷ به حدود ۰/۳۵ دلار در سال ۲۰۰۰ و قیمت باور نکردنی ۰/۰۸ دلار در سال ۲۰۰۱(بر حسب دلار سال ۱۹۷۷) سقوط کرد. هرنسل با فرض قیمت ثابت دلار در طول عمر خود و با فاکتور ۱۰ تا ۳۰ ارزان شده است. از سال ۱۹۹۶ زیاد شدن تعداد تولید کنندگان باعث کاهش مرزها شده و باعث افزایش نرخ سقوط قیمت و در نهایت کاهش قیمت DRAM شد. در زمانهایی که تقاضا بیش از تولید بود مانند سالهای ۸۸ – ۱۹۸۷ و ۹۳ – ۱۹۹۲ بطور موقت قیمت‌ها افزایش یافت و باعث کم شده نرخ کاهش شد. در اواخر سال ۲۰۰۰ و سال ۲۰۰۱ ، تولید اضافه بود و باعث کاهش بیشتر قیمت ها شد ولی این شرایط همیشگی نخواهد بود.

قیمت ریزپردازنده نیز در طول زمان کاهش یافت ولی چون آنها مانند DRAM استاندارد نبودند، ارتباط بین هزینه و قیمت پیچیده‌تر می‌باشد. در دوره‌هایی که رقابت شدید بود، قیمت‌ها به هزینه‌ها

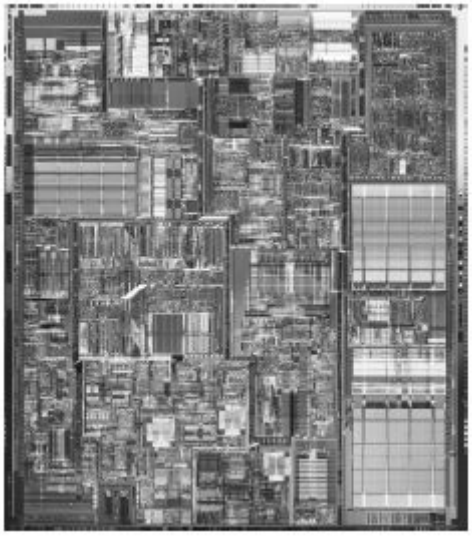
نزدیک بودند، گر چه بعید است که تولید کنندگان، ریزپردازنده‌ها را با ضرر به فروش برسانند. شکل شماره ۶ نمودار قیمت ریزپردازنده پنتیوم ۳ را نشان می‌دهد.




شکل ۶ – نمودار قیمت ریزپردازنده پنتیوم ۳

قیمت ریزپردازنده پنتیوم ۳ با یک فرکانس خاص در طول زمان کاهش می‌یابد چون بهبود کیفیت تولید باعث کاهش هزینهٔ اجزا شده و قیمت ها را کاهش می‌دهد. داده ها از گزارش ریزپردازنده چاپ ماه می ۲۰۰۰ گرفته شده است. ریزپردازندهایی که اخیراً عرضه شده‌اند آنقدر ارزان می‌شوند که تقریباً به پایین‌ترین حد خود برسند، این حد پایین امروزه ۱۰۰ تا ۲۰۰ دلار می‌باشد. کاهش قیمت با فرض یک محیط رقابتی می‌باشد که در آن قیمت ارتباط نزدیکی با هزینه‌ها دارد.

حجم تولید عامل دومی‌است که مشخص کننده هزینه می‌باشد. افزایش حجم تولید از چند جهت باعث کاهش هزینه می‌شود. ابتدا، باعث کاهش زمان مورد نیاز برای سقوط منحنی یادگیری که متناسب با تعداد سیستم ها ( یا تراشه های) تولید شده است، می‌باشد. دومین دلیل، حجم تولید باعث کاهش هزینه‌ها می‌شود چون باعث افزایش فروش و افزایش سود دهی می‌شود. بطور سرانگشتی بعضی از طراحان تخمین می‌زنند که با دو برابر شدن میزان تولید، هزینه‌ها ۱۰٪ کاهش می‌یابد.



شکل ۷ – شمای داخلی ریزپردازنده پنتیوم ۴ (مربوط به شرکت اینتل)

همچنین حجم تولید باعث کاهش هزینه تولید به ازای هر کالای تولیدی شده و باعث می‌شود که هزینه و قیمت به هم نزدیکتر شوند. در صفحات بعدی به فاکتورهای دیگری که روی قیمت فروش تأثیر دارند اشاره خواهد شد.

کالاهای تجاری (Commodities)، محصولاتی هستند که توسط چندین تولید کننده و در حجم بالایی فروخته می‌شوند و اصولاً یکسان هستند. بطور مثال تمام اجناس موجود در قفسه‌های خواربار فروشی‌ها از این نوع هستند. همچنین حافظه‌های استاندارد DRAM، دیسک‌ها، مانیتورها و صفحه کلید از این نوع هستند. در ۱۰ سال گذشته اکثر تجارت سطح پایین کامپیوتر تجارت این اجزا و ساخت کامپیوترهای شخصی همساز با آی بی ام بوده است. تعدادی تولید کننده هستند که در اصل محصولات یکسانی را عرضه می‌کنند و شدیداً در رقابت هستند. که این رقابت باعث کاهش فاصلهٔ بین هزینه تولید و قیمت فروش و همچنین کاهش هزینه تولید می‌شود. این کاهش قیمت‌ها به این دلیل است که بازار چیزهای تجارتی حجم زیادی داشته و تعریف روشنی از محصولات وجود دارد و باعث می‌شود که

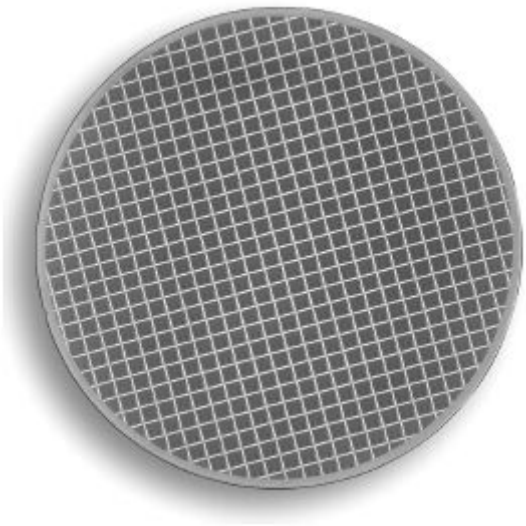
چندین تولید کننده در ساخت یک محصول با هم رقابت کنند. در نتیجه هزینه کلی تولید کاهش می‌یابد چون رقابت بین تولید کنندگان و حجم زیاد تولید باعث می‌شوند که عرضه کنندگان به بهترین قیمت برسند. این باعث می‌شود که تجارت سطح آخر کامپیوتر بتوانند ترکیب بهتری از کارآیی – قیمت را عرضه کند و باعث رسیدن به قیمت‌های مناسب‌تر در مقایسه با بقیه قسمت‌ها شده و مانند هر محصول تجاری دیگر باعث رشد سریع در بازار نهایی البته با سود کم شود.

## هزینه یک مدار مجتمع

چرا باید یک کتاب معماری کامپیوتر بخشی دربارهٔ هزینهٔ یک مدار مجتمع داشته باشد؟ در بازار پررقابت کامپیوتر که اجزای استاندارد مانند دیسک و حافظه DRAM قسمت عمده‌ای از هزینه یک سیستم را تشکیل می‌دهند، هزینه مدار مجتمع بخش عمده‌ای از هزینه‌ای را تشکیل می‌دهد که به خصوص در حجم زیاد و بازار حساس به قیمت، بین کامپیوترهای مختلف متفاوت می‌باشد. بنابراین طراحان کامپیوتر باید با هزینهٔ تولید تراشه آشنا باشند تا درک درستی از هزینه کامپیوترهای امروزی داشته باشند.

گرچه هزینه تولید مدارات مجتمع بصورت نمایی سقوط می‌کند، ولی فرآیند اصلی تولید نیمه هادی سیلیکون تغییری نکرده است. یک ویفر هنوز باید تست شده سپس هر هسته جدا شده و بسته بندی شود. (شکل‌های ۷ و ۸ را ببینید). بنابراین هزینه یک مدار مجتمع بسته بندی شده با رابطه زیر محاسبه می‌شود:

$$\text{هزینه یک‌مدار مجتمع} = \frac{\text{هزینه هسته} + \text{هزینه تست هسته} + \text{هزینه تست نهائی و بسته بندی}}{\text{تعداد مدارات موفق در تست}}$$



شکل۸ – شمای یک ویفر

این ویفر ۸ اینچی حاوی ۵۶۴ پردازندهٔ MIPS64 R20K که با فرآیند ۰/۱۸ میکرون تولید شده است. R20K یک پیاده سازی از معماری MIPS64 با مجموعه دستورالعمل‌های گسترش یافته می‌باشد که MIPS-3D نامیده شده و برای استفاده در محاسبات گرافیکی ۳ بعدی استفاده می‌شود. در سرعت‌های بین ۵۰۰ تا ۷۵۰ مگاهرتز در دسترس بوده و قادر به اجرای دو عمل صحیح در یک کلاک می‌باشد. با استفاده از دستور العمل‌های MIPS-3D، پردازنده R20K می‌تواند تا ۳ میلیارد عمل ممیز شناور را در ثانیه انجام دهد.( مربوط به شرکت فناوری MIPS)

در این بخش، روی قیمت هسته متمرکز شده و بطور مختصر به تست و بسته بندی پرداخته می‌شود. توضیح بیشتر هزینه‌های تست و بسته بندی در تمرینات آمده است.

برای یادگیری روش پیش بینی تعداد تراشه‌های خوب در یک ویفر باید در ابتدا بدانیم که ویفر چند هسته، گنجایش دارد و سپس یادبگیریم که چگونه حدس بزنیم که چند درصد از آنها کار خواهند کرد. از اینجا می‌توان هزینه را بصورت زیر حدس زد:

$$\text{هزینه هسته} = \frac{\text{هزینه ویفر}}{\text{درصد هسته‌های سالم} \times \text{تعداد هسته‌در ویفر}}$$

جالب ترین ویژگی این فرمول اولیه برای محاسبهٔ هزینه، وابستگی آن به‌اندازهٔ هسته می‌باشد.

تعداد هسته‌ها در یک ویفر در اصل مساحت ویفر تقسیم بر مساحت هسته می‌باشد. بنابراین یک حدس دقیقتر بصورت زیر است:

$$\text{تعداد هسته‌در یك ویفر} = \frac{\pi \times \left(\frac{\text{قطر ویفر}}{2}\right)^2}{\text{مساحت هسته}} - \frac{\pi \times \text{قطر ویفر}}{\sqrt{2 \times \text{مساحت هسته}}}$$

اولین جزء فرمول فوق نسبت مساحت ویفر به مساحت هسته می‌باشد. دومین جزء این فرمول مربوط به قرار دادن یک سری مربع در یک دایره می‌باشد، چون هسته‌های مربع شکل در ویفر دایره شکل قرار دارند. تقسیم محیط به قطر یک مربع هسته تقریباً تعداد هسته‌های موجود در لبه ها را نشان می‌دهد. مثلاً یک ویفر با قطر ۳۰ سانتیمتر (۱۲ اینچی) تعداد (۱/۴۱ ÷ ۳۰ * π ) – ۲۲۵ * π یعنی ۶۴۰ هسته یک سانتیمتری را در خود جای می‌دهد.

---

مثال: تعداد هسته‌های موجود در یک ویفر ۳۰ سانتیمتری برای هسته‌هایی که طول و عرض آنها ۰/۷ سانتیمتر می‌باشد را محاسبه کنید.

پاسخ: مساحت هسته ۰/۴۹ سانتیمتر مربع می‌باشد، پس:

$$\text{تعداد هسته‌در یك ویفر} = \frac{\pi \times \left(\frac{30}{2}\right)^2}{0.49} - \frac{\pi \times 30}{\sqrt{2 \times 0.49}} = \frac{706.3}{0.49} - \frac{94.2}{0.99} = 1347$$

---

ولی این رابطه فقط حداکثر تعداد هسته ها را در یک ویفر مشخص می‌کند. سؤال اساسی این است که چه نسبتی یا چند درصد از هسته ها در یک ویفر سالم هستند، یا میزان موفقیت هسته چیست؟ یک مدل ساده از میزان موفقیت مدارات مجتمع، فرض می‌کند که خرابی بطور تصادفی در سطح ویفر توزیع شده و میزان موفقیت با پیچیدگی فرآیند تولید نسبت عکس دارد:

$$\text{درصد موفقیت ویفر} = \text{درصد موفقیت هسته} \times \left(1+\frac{\text{مساحت هسته} \times \text{خرابی در واحد سطح}}{\alpha}\right)^{-\alpha}$$

که در آن درصد موفقیت ویفر نشان دهندهٔ ویفرهایی است که کاملاً خراب بوده و نیاز به تست ندارد. برای سادگی فرض می‌کنیم موفقیت ویفر ۱۰۰٪ باشد. خرابی در واحد سطح‌اندازه‌گیری تصادفی خرابیهایی است که در تولید رخ می‌دهد. در سال ۲۰۰۱، این عامل ۰/۴ تا ۰/۸ در سانتیمتر مربع می‌باشد و بستگی به سابقهٔ فرآیند دارد (منحنی یادگیری که قبلاً دربارهٔ آن صحبت شد.) در نهایت α پارامتری است که نسبت معکوس با تعداد سطوح ماسک‌ها داشته و میزان پیچیدگی فرآیند را نشان می‌دهد و برای موفقیت هسته‌ها حیاتی می‌باشد.

---

مثال: میزان موفقیت هسته‌هایی را که طول و عرض آنها ۱cm و هسته‌هایی را که طول و عرض آنها 7/0 cm هست را بدست آورید. فرض کنید چگال خرابی در $cm^2$ برابر ۰/۶ می‌باشد.

پاسخ: مساحت هسته‌ها $1\ cm^2$ و $0.49\ cm^2$ می‌باشد. برای هسته‌های بزرگتر:

$$\text{درصد موفقیت هسته} = \left(1+\frac{0.6\times 1}{4}\right)^{-4} = 0.57$$

و برای هسته‌های کوچکتر:

$$\text{درصد موفقیت هسته} = \left(1+\frac{0.6\times 0.49}{4}\right)^{-4} = 0.75$$

---

مخرج کسر مربوط به هزینه هسته، تعداد هسته‌های سالم در یک ویفر می‌باشد که از ضرب تعداد کل هسته در یک ویفر در درصد موفقیت هسته بدست می‌آید. (که در ارتباط با میزان خرابی‌ها می‌باشد.) مثال فوق پیش بینی می‌کند که ۳۶۶ هسته $1cm^2$ یا ۱۰۱۴ ویفر $49/0\,cm^2$ در یک ویفر با قطر 30 cm باشد. اکثر ریزپردازنده‌های ۳۲ و ۶۴ بیتی در فناوری مدرن ۰/۲۵ میکرونی بین این دو اندازه هستند. بعضی از پردازنده‌ها ابتدا قبل از نازک شدن $2\,cm^2$ هستند. پردازنده‌های ۳۲ بیتی جاسازی شده تا

کوچکی $25/0\,cm^2$ نیز هستند. ولی پردازنده‌های کنترلرهای جاسازی شده (مثلاً برای چاپگرها یا اتومبیل‌ها) کمتر $1/0\,cm^2$ هستند. شکل ۳۴ برای تمرین ۱–۸ اندازهٔ هسته و فناوری را برای تعدادی از پردازنده‌های امروزی نشان می‌دهد.

با توجه به فشار زیادی که روی قیمت محصولات تجاری مانند DRAM یا SRAM وجود دارد، طراحان از افزونگی برای بالا بردن کارآیی استفاده می‌کنند. طی سالهای زیادی حافظه‌های پویا معمولاً دارای افزونگی در سلولهای حافظه بودند تا بتوانند خرابی بعضی از سلولها را تحمل کنند. طراحان حافظه‌های ایستا هم در حافظه‌های ایستای استاندارد و هم در حافظه‌های نهان ایستا در ریزپردازنده‌ها از همین ایده استفاده می‌کنند. مشخص است که وجود افزونگی باعث رسیدن به هدف می‌شود.

پردازش یک ویفر با قطر ۳۰ سانتیمتر با فناوری پیشرفته با ۴ تا ۶ لایهٔ فلز در سال ۲۰۰۱ بین ۵ تا ۶ هزار دلار هزینه دارد. با فرض هزینه پردازش ۵۵۰۰ دلاری برای یک ویفر نیمه هادی، هزینه مربوط به هسته یک پردازنده با مساحت $49/0\,cm^2$ حدود ۵/۴۲ دلار و برای یک هسته با مساحت $1cm^2$ حدود ۱۵/۰۳ دلار یعنی سه برابر می‌باشد در صورتی که مساحت آن تقریباً دو برابر می‌باشد.

یک طراح کامپیوتر باید چه نکاتی را در رابطه با هزینه تراشه‌ها در نظر داشته باشد؟ فرآیند مساحت، هزینه ویفر نیمه هادی، کیفیت ویفر و تعداد خرابی در واحد سطح را تعیین می‌کند، پس قسمتی که در کنترل طراح می‌باشد مساحت هستهٔ تراشه می‌باشد. چون برای فرآیندهای پیشرفته امروزی $\alpha$حدود ۴ می‌باشد، پس هزینه تولید هسته با توان چهارم اندازهٔ آن افزایش می‌یابد. در عمل، چون تعداد خرابی در واحد سطح کم می‌باشد، باعث افزایش تعداد هسته‌های سالم در یک ویفر شده و سبب می‌شود که تقریباً هزینه به نسبت مربع مساحت هسته باشد. طراح کامپیوتر مساحت را تعیین می‌کند بنابراین روی هزینه اثر

می‌گذارد این اثر می‌تواند از طریق حذف یا اضافه کردن یک واحد به هسته یا با تغییر تعداد پایه‌های ورودی و خروجی باشد.

قبل از اینکه یک قطعه برای استفاده در کامپیوتر آماده شود، هسته باید آزمایش شود (تا هسته‌های معیوب جدا شوند)، سپس هسته بسته بندی شده و مجدداً تراشه آماده شده آزمایش می‌شود. این مراحل هزینه‌های زیاد دیگری را اضافه می‌کنند. این فرآیند و اثری که آنها روی هزینه‌ها دارند در تمرین ۱-۸ بحث و بررسی می‌شود.

بررسی فوق برای مشخص کردن هزینه‌های تولید یک هستهٔ تراشه برای مدارات مجتمع تولید انبوه بود. ولی به هر حال هزینه ثابت دیگری نیز وجود دارد که هزینه تولید مدارات مجتمع با تعداد کم( یعنی کمتر از یک میلیون قطعه) را تحت تأثیر قرار می‌دهد که آن هزینه مجموعهٔ ماسک ها می‌باشد. هر مرحله در فرآیند مدارات مجتمع نیاز به یک ماسک جداگانه دارد. بنابراین برای فرآیند تولید مدارات مدرن با تجمع بالا که ۴ تا ۶ لایهٔ فلز دارند، هزینه تولید ماسک از یک میلیون دلار نیز بیشتر می‌باشد. مشخص است که این هزینهٔ ثابت و زیاد روی هزینه‌های نمونه سازی و عیب یابی اثر گذاشته و برای محصولات با تعداد کم تأثیر زیادی در هزینه نهایی دارد. چون هزینه ماسک در حال افزایش می‌باشد، طراح ممکن است از مدارات قابل برنامه ریزی مجدد استفاده کند تا قطعه قابلیت انعطاف داشته باشد یا ممکن است از آرایه‌های گیت قابل برنامه ریزی استفاده کند،( این آرایه ها تعداد ماسک‌های کمتری دارند) و به این ترتیب هزینه ماسک ها کاهش می‌یابد.

## توزیع هزینه لایه‌های یک سیستم : یک مثال

برای تصور درست هزینه نیمه‌هادی در شکل ۹ هزینه‌های اجزای مختلف یک کامپیوتر شخصی هزار دلار در سال ۲۰۰۱ نشان داده شده است. گرچه هزینه بعضی از قسمت‌های این ماشین با گذشت

زمان کاهش می‌یابد ولی بعضی از اجزا مانند منبع تغذیه یا بدنهٔ اصلی، امکان کمی‌برای کاهش هزینه‌ها دارند. همچنین انتظار داریم که ماشین‌های آینده حافظهٔ زیاد و فضای دیسک بزرگی داشته باشند، بنابراین قیمت یک کامپیوتر با پیشرفت فناوری به میزان کمی کاهش می‌یابد.

| System | Subsystem | Fraction of total |
|---|---|---|
| Cabinet | Sheet metal, plastic | 2% |
| | Power supply, fans | 2% |
| | Cables, nuts, bolts | 1% |
| | Shipping box, manuals | 1% |
| | **Subtotal** | **6%** |
| Processor board | Processor | 22% |
| | DRAM (128 MB) | 5% |
| | Video card | 5% |
| | Motherboard with basic I/O support, networking | 5% |
| | **Subtotal** | **37%** |
| I/O devices | Keyboard and mouse | 3% |
| | Monitor | 19% |
| | Hard disk (20 GB) | 9% |
| | DVD drive | 6% |
| | **Subtotal** | **37%** |
| Software | OS + Basic Office Suite | 20% |

شکل ۹ – تخمین توزیع هزینه‌های اجزای یک کامپیوتر شخصی هزار دلاری در سال۲۰۰۱

توجه کنید که بیشترین هزینه یک جزء را پردازنده داشته و بعد از آن مانیتور قرار داد.(ولی در سال ۱۹۹۵ بیشترین قیمت را حافظه‌های پویا ( داشته که تقریباًیک سوم هزینهٔ یک سیستم را شامل می‌شدند. ولی پس از آن هزینه یک مگابیت حافظه با نسبت ۱۵ کاهش یافت.) توما (Touma) در سال ۱۹۹۳ هزینه و قیمت اجزای کامپیوتر را به طور مفصل شرح داده است. تخمین‌های شکل فوق با توجه به قیمت عمده‌فروشی اجزای مختلف می‌باشد.

## رابطه قیمت و هزینه، و دلایل و میزان اختلاف آنها

طراح هزینهٔ قطعات را در نظر می‌گیرد ولی این هزینه‌ها با قیمت نهایی که مصرف کننده پرداخت می‌کند تفاوت زیادی دارد. ولی چگونه یک طراح کامپیوتر باید اطلاعات قیمت را در نظر بگیرد؟ هزینه‌ها بعد از طی چندین مرحله به قیمت تبدیل می‌شود و یک طراح کامپیوتر باید بداند که چگونه

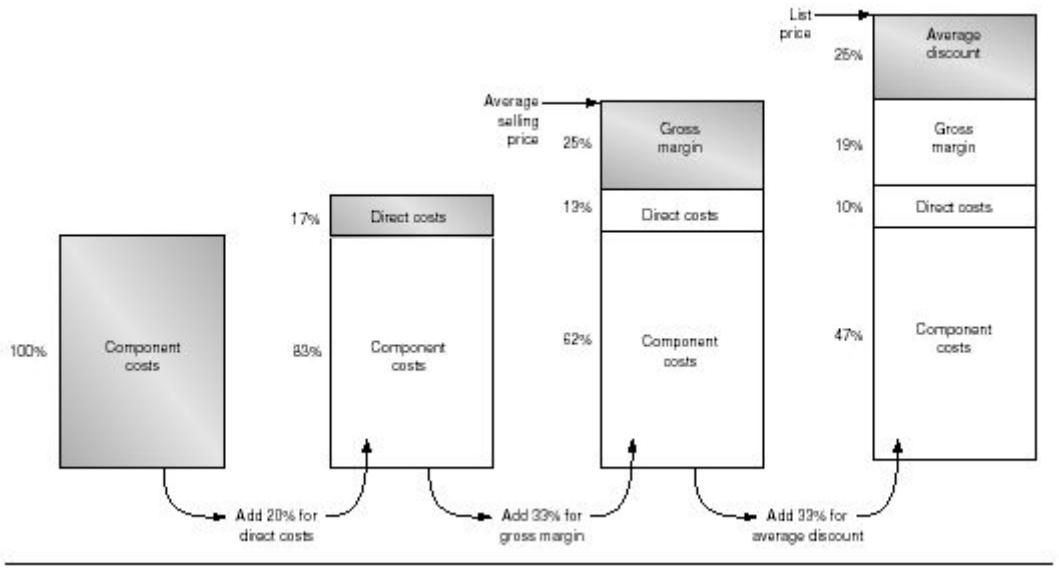
قیمت فروش تعیین می‌شود. مثلاً هزار دلار افزایش هزینه ممکن است باعث افزایش ۳ تا ۴ هزار دلار در قیمت نهایی شود. بدون در نظر گرفتن رابطه هزینه و قیمت، یک طراح کامپیوتر نمی‌تواند دربارهٔ حذف، اضافه کردن یا تعویض یک قطعه تصمیم بگیرد.

رابطهٔ بین قیمت و تعداد باعث افزایش اثر قیمت در اثر تغییر هزینه‌ها بخصوص برای مصرف کننده نهایی می‌شود. افزایش قیمت کامپیوتر باعث کاهش میزان فروش می‌شود. در نتیجهٔ کاهش میزان فروش باعث افزایش هزینه‌ها شده و قیمت باز هم افزایش می‌یابد. بنابراین تغییر اندکی در هزینه‌ها می‌تواند اثر زیادی داشته باشد. رابطه بین هزینه‌ها و قیمت پیچیده بوده و کتابهای زیادی در این زمینه نوشته شده است. هدف این بخش ارائهٔ مقدمهٔ ساده ای از عواملی که روی قیمت تأثیر دارند و میزان تأثیر این عوامل است.

دسته بندی عوامل افزایش قیمت را می‌توان به عنوان عوارض برای هزینه‌ها یا درصدی از قیمت انجام داد. به این اطلاعات از هر دو جنبه نگاه خواهد شد. تفاوت بین هزینه و قیمت همچنین به موقعیت فروش یک شرکت در بازار بستگی دارد. برای نمایش این تفاوت ها در شکل ۱۰ تفاوت بین هزینه مواد اولیه و قیمت مصرف کننده نشان داده شده است در این شکل قیمت از چپ به راست با توجه به هزینه‌های بالا سری، اضافه می‌شود.

هزینه مستقیم، هزینه‌هایی است که مستقیماً برای تولید یک محصول پرداخت می‌شود. این هزینه شامل دستمزد، هزینه خرید، ضایعات (بخشی که کنار گذاشته می‌شوند) و ضمانت، یعنی اجزائی که در مدت ضمانت مشتری خراب می‌شوند، می‌باشد. هزینه‌های مستقیم معمولاً ۱۰ تا ۳۰ درصد به هزینه اجزا

اضافه می‌کند. هزینه تعمیر و نگهداری منظور نشده است چون معمولاً این هزینه‌ها توسط مصرف کننده پرداخت می‌شود، البته هزینه‌های مربوط به ضمانت در اینجا با هزینه‌های ناخالص منظور می‌شود.




شکل ۱۰ تقسیم بندی قیمت برای یک کامپیوتر شخصی هزار دلاری

هر افزایشی مانند یک مالیات بر قیمت قبلی نشان داده شده است. درصدهای قیمت جدید در سمت چپ هر ستون نشان داده شده است

هزینه‌های ناخالص، هزینه بالاسری یک شرکت که نمی‌توان آنها را مستقیماً به یک محصول خاص نسبت داد، می‌باشد. به این هزینه‌ها می‌توان هزینه‌های غیر مستقیم نیز گفت. این هزینه‌ها شامل هزینه‌های تحقیق و توسعه، بازاریابی، فروش، نگهداری تجهیزات تولید، هزینه‌های ساختمان‌ها، هزینهٔ بورس، عوارض و مالیات می‌باشد. با اضافه کردن هزینه قطعات به هزینه‌های مستقیم و هزینه‌های ناخالص قیمت متوسط فروش یا قیمت عمده فروشی بدست می‌آید. این مبلغ در واقع پولی است که به ازای هر محصولی که فروخته می‌شود، مستقیماً به تولید کننده پرداخت می‌شود. هزینه‌های ناخالص معمولاً ۱۰٪ تا ۴۵٪ از

قیمت عمده فروشی را تشکیل می‌دهند که به وابسته به نوع محصول می‌باشد. تولید کنندگان کامپیوترهای شخصی به دلایل مختلف هزینه‌های نا خالص خیلی کمی دارند.

دلیل اول این است که هزینه تحقیق و توسعه خیلی کمی دارند. دلیل دوم پایین بودن هزینه فروش آنها می‌باشد چون آنها معمولاً توزیع غیر مستقیم (از طریق پست، اینترنت، تلفن یا فروشگاه‌های زنجیره‌ای) به جای بازاریاب‌ها، دارند. دلیل سوم این است که چون محصول آنها انحصاری نمی‌باشد، رقابت در آن زیاد بوده و این رقابت باعث پایین آمدن قیمت و سود می‌شود که باعث کاهش هزینه‌های ناخالص می‌شود.

قیمت مصرف کننده با قیمت عمده فروشی متفاوت می‌باشد، چون معمولاً شرکت ها تخفیف عمده فروشی دارند که باعث کم شدن متوسط قیمت فروش می‌شود. از زمانی که کامپیوترهای شخصی تبدیل به یک کالای تجاری شده است ، فروشندگان قیمت ها را مقدار زیادی کاهش داده‌اند و این کار باعث شده است که قیمت مصرف کننده و عمده فروشی به یکدیگر نزدیک شوند.

همان طور که گفته شد، قیمت‌ها به رقابت حساس هستند. یک شرکت ممکن است قادر نباشد که محصولات خود را به قیمتی بفروشد که هزینه‌های ناخالص را شامل شود. در بدترین حالت ، قیمت‌ها باید به مقدار زیادی کاهش یابد و از هزینه‌های ناخالص نیز پایین‌تر بیاید و باعث منفی شدن سود شود. شرکتی که می‌خواهد سهم بیشتری از بازار را داشته باشد باید قیمت‌ها و سود را کاهش دهد تا جذابیت محصولات آن شرکت افزایش یابد. اگر میزان فروش افزایش یابد، هزینه‌ها کاهش می‌یابد. به خاطر داشته باشید که این رابطه بسیار پیچیده بوده و برای توضیح کامل آن به یک کتاب کامل نیاز می‌باشد و بیان آن

در یک بخش از یک فصل یک کتاب ممکن نیست. مثلاً اگر یک شرکت قیمت‌ها را کاهش دهد ولی نتواند میزان فروش محصولات خود را افزایش دهد، ضرر زیادی خواهد داشت.

خیلی از مهندسین تعجب می‌کنند که چرا اکثر شرکت ها فقط ۴٪ ( برای کامپیوترهای شخصی) تا ۱۲٪ (برای سرویس دهنده‌ها) از درآمد خود را صرف تحقیق و توسعه می‌کنند، که این شامل تمام هزینه‌های مهندسی می‌شود (به جز هزینه‌های تولید وحقوق مهندسین تولیدی). این درصد مشخص در گزارشات سالیانه شرکت ها منعکس شده و در نشریات کشوری چاپ می‌شود، بنابراین بعید است که این درصد در آینده نیز تغییر کند. در حقیقت تجربه نشان داده است که شرکت هایی که بیش از ۱۵ تا ۲۰ درصد را صرف تحقیق و توسعه می‌کنند به ندرت در بلند مدت موفق بوده‌اند.

اطلاعات فوق مشخص می‌کند که شرکت‌ها هزینه‌های بالاسری را به هزینه اضافه می‌کنند تا قیمت محصولات مشخص شود و این مسئله در مورد اکثر شرکت‌ها صادق می‌باشد. نکته دیگر سرمایه گذاری برای تحقیق و توسعه می‌باشد. سرمایه گذاری ۴٪ تا ۱۲٪ از درآمد برای تحقیق و توسعه به معنی آن است که برای سرمایه‌گذاری یک میلیون دلار برای تحقیق و توسعه لازم است که ۸ تا ۲۵ میلیون دلار فروش وجود داشته باشد. بنابراین باید هزینه‌های ناخالص متناسب با هر محصول و میزان فروش آن محصول در نظر گرفته شود.

کامپیوترهای بزرگ و گران قیمت معمولاً هزینه‌های زیادی برای طراحی نیاز دارند، هزینه این کامپیوترها ۱۰ برابر هزینه ساخت می‌باشد که مقداری از این هزینه مربوط به طراحی می‌باشد چون ماشین‌های بزرگ و گران قیمت به تعداد کمی‌فروخته می‌شوند باید هزینه‌های ناخالص برای آنها زیاد باشد تا سرمایه گذاری انجام شده به شرکت برگردانده شود. پس ماشین‌های بزرگ با دو مسئله درگیر

هستند یکی اینکه میزان فروش آنها کم و دیگر اینکه هزینه تحقیق و توسعه آنها زیاد می‌باشد بنابراین نسبت قیمت به هزینه برای آنها بیشتر از ماشین‌های کوچک می‌شود.

موضوع هزینه و هزینه – کارآیی پیچیده می‌باشد. برای یک طراح کامپیوتر فقط یک هدف وجود ندارد. از یک طرف برای طرح‌های با کارآیی بالا هزینه اهمیت ندارد، سوپر کامپیوترها از این دسته می‌باشند ولی سهم این نوع کامپیوترها یعنی کامپیوترهایی که فقط در آنها کارآیی مهم می‌باشد، از بازار کم می‌باشد. در طرف دیگر طرح‌های ارزان قیمت هستند که در آنها کارایی فدای هزینهٔ کمتر شده است. مثلاً بازار کامپیوترهای جاسازی شده مانند پردازنده‌های تلفن همراه از این نوع هستند. در بین این دو نوع طراحی، طرح‌هایی هستند که در آنها هزینه و کارآیی باید متناسب باشند، یعنی طراح باید بین هزینه‌ها و کارآیی توازن برقرار کند. کامپیوترهای شخصی، ایستگاه‌های کاری و سرویس دهنده‌ها (حداقل سرویس دهنده‌های کوچک و متوسط) از این نوع می‌باشند.

در ۱۰ سال گذشته با کوچک شدن کامپیوترها، طرح‌های کم هزینه‌ها و طرحهای با تناسب هزینه و کارآیی اهمیت یافته‌اند. در این بخش در مورد هزینه‌ها بحث شد در بخش بعدی در مورد کارآیی صحبت می‌شود.

## ۱ – ۵ اندازه گیری و گزارش کارآیی

وقتی گفته می‌شود که یک کامپیوتر از کامپیوتر دیگر سریعتر می‌باشد، منظور چیست؟ یک کاربر کامپیوتر شخصی ممکن است بگوید که کامپیوتری سریعتر است که برنامه را در زمان کمتری اجرا کند، ولی مدیر یک مرکز کامپیوتری که یک سیستم سرویس دهندهٔ بزرگ دارد می‌گوید کامپیوتری سریعتر است که در یک ساعت بتواند تعداد بیشتری کار را به انجام برساند. یک کاربر کامپیوتر مایل است که زمان پاسخ گویی یعنی زمان بین شروع و خاتمهٔ یک کار کاهش یابد، به این مدت، زمان اجرا نیز

می‌گویند، مدیر یک مرکز بزرگ پردازش داده مایل است که بازدهٔ سیستم یعنی کار انجام شده در واحد زمان زیاد باشد. برای مقایسهٔ طرح‌های مختلف لازم است که کارآیی دو کامپیوتر مختلف، که آنها را X و Y می‌نامیم، با هم مقایسه شوند. عبارت کامپیوتر X از کامپیوتر Y سریعتر می‌باشد، به این معنی است که زمان پاسخ گویی یا زمان اجرا در کامپیوتر X از Y برای یک کار خاص کمتر می‌باشد. به همین ترتیب عبارت کامپیوتر X از کامپیوتر Y به مقدار n بار سریع ترمی‌باشد" به این معنی است:

$$\frac{\text{زمان اجرای } Y}{\text{زمان اجرای } X} = n$$

چون زمان اجرا با کارآیی نسبت عکس دارد، رابطهٔ زیر بدست می‌آید :

$$n = \frac{\text{زمان اجرای } Y}{\text{زمان اجرای } X} = \frac{\frac{1}{\text{کارآئی } Y}}{\frac{1}{\text{کارآئی } X}} = \frac{\text{کارآئی } X}{\text{کارآئی } Y}$$

عبارت " بازده کامپیوتر X، برابر ۱/۳ بازده Y می‌باشد" ، به این معنی است که تعداد کارهای انجام شده در واحد زمان روی کامپیوتر X برابر ۱/۳ تعداد کارهای انجام شده روی کامپیوتر Y می‌باشد. چون کارآیی و زمان اجرا معکوس یکدیگر هستند، افزایش کارآیی باعث کاهش زمان اجرا می‌شود. برای جلوگیری از ایجاد ابهام در استفاده از کلمات افزایش و کاهش معمولاً از عبارت "بهبود کارآیی" یا "بهبود زمان اجرا" به جای بیان افزایش کارآیی یا کاهش زمان اجرا استفاده می‌شود.

برای ما بازده مهم می‌باشد یا زمان پاسخ گویی، عامل مهم زمان می‌باشد. کامپیوتری که مقدار مشخصی از کار را در زمان کمتری انجام دهد سریعتر می‌باشد تفاوت وقتی است که انجام یک کار (زمان پاسخ گویی) با انجام چند کار (بازده) مقایسه می‌شود. ولی زمان تنها عامل ارزیابی برای مقایسه

کارآیی کامپیوترها نیست. معیارهای دیگری برای ارزیابی کارآیی کامپیوترها وجود دارد، این معیارها به خوبی قابل فهم بوده و بطور عمومی‌می‌تواند کارآیی کامپیوتر را اندازه گیری کند ولی این معیارها ممکن است سرویسی را اندازه گیری کنند که کامپیوتر هیچگاه آن را اجرا نمی‌کند. پس معیار اندازه‌گیری مناسب و مطمئن برای کارآیی، زمان اجرای برنامه‌های واقعی می‌باشد و معیارهای دیگر غیر از معیار زمان باعث رسیدن به نتایج غلط و اشتباه طراحان کامپیوتر خواهد شد. بعضی از خطرات معیارهای دیگر در بخش ۱ – ۹ نشان داده خواهد شد.

## اندازه گیری کارآیی

حتی زمان اجرا نیز به روش‌های مختلفی تعریف می‌شود و بستگی به نوع محاسبه دارد. سرراست‌ترین تعریف از زمان اجرا، زمان ساعت دیواری(Wall-Clock time) یا زمان پاسخ یا زمان سپری شده که تأخیر مورد نیاز برای تکمیل یک کار یا برنامه می‌باشد، این زمان شامل دسترسی به دیسک، دسترسی به حافظه، فعالیت‌های ورودی و خروجی، سربارهای سیستم عامل و هر چیز دیگر می‌باشد. در حالت چند برنامگی زمانی که پردازنده منتظر ورودی یا خروجی یک برنامه می‌باشد، روی یک برنامه دیگر کار می‌کند و نیازی ندارد که زمان صرف شده برای یک برنامه را حداقل کند. حال نیاز به روشی برای در نظر گرفتن این فعالیت ها داریم. زمان پردازنده این مسئله را در نظر می‌گیرد، این زمان فقط زمانی است که پردازنده محاسبات را انجام می‌دهد و شامل زمان‌های ورودی و خروجی و اجرای دیگر برنامه‌ها نمی‌شود.(در حقیقت زمان پاسخ زمانی است که توسط کاربر مشاهده می‌شود و با زمانی که پردازنده مصرف می‌کند متفاوت می‌باشد.) زمان پردازنده را می‌توان به زمانی که صرف برنامه می‌شود و زمان پردازنده برای کاربر نامیده می‌شود و زمانی که صرف سیستم عامل .و انجام وظایف آن می‌شود و زمان پردازنده برای سیستم نامیده می‌شود، تقسیم کرد.

این زمان‌های مجزا را در سیستم عامل یونیکس با دستور time می‌توان مشاهده کرد. مثلاً خروجی این دستور برای اجرای یک برنامه می‌تواند به صورت زیر باشد:

90.7u 12.9s 2:39 65%

یعنی اینکه زمان مربوط یه کاربر ۹۰/۷ ثانیه بوده و زمان مربوط به سیستم عامل ۱۲/۹ ثانیه بوده و کل زمان صرف شده ۲ دقیقه و ۳۹ ثانیه(۱۵۹ ثانیه) و درصد زمانی که در اختیار برنامه بوده ۱۵۹/(۱۲/۹+ ۹۰/۷) یا ۶۵ درصد می‌باشد. بیش از یک سوم زمان سپری شده منتظر ورودی و خروجی بوده یا برنامه‌های دیگر در حال اجرا بوده‌اند. اکثر اندازه‌گیری‌ها زمان مربوط به سیستم عامل را نادیده می‌گیرند چون معمولاً سیستم عامل زمان مربوط به خودش را به دقت اندازه گیری نمی‌کند (این اشکال مربوط به سیستم عامل یونیکس می‌باشد.) و در نظر گرفتن زمان مربوط به سیستم عامل باعث اشتباه در اندازه گیری کارآیی با توجه به سیستم عامل‌های مختلف می‌شود. از طرف دیگر برنامه سیستم در یک ماشین ممکن است برنامهٔ کاربر در یک سیستم دیگر باشد، همچنین هیچ برنامه‌ای بدون وجود سیستم عامل قابل اجرا نمی‌باشد بنابراین مناسب است که مجموع زمان پردازنده برای کاربر و زمان پردازنده برای سیستم را در نظر گرفت.

در مطالب بیان شده تفاوت بین زمان سپری شده و زمان پردازنده گفته شد. عبارت کارآیی سیستم به زمان سپری شده در یک سیستم بدون بار اشاره می‌کند در صورتی که کارآیی پردازنده زمان مربوط به کاربر در یک سیستم بار نشده می‌باشد. در این فصل کارآیی پردازنده بررسی می‌شود، گرچه به کارآیی با توجه به زمان سپری شده نیز توجه می‌شود.

## انتخاب برنامه هایی برای ارزیابی کارآیی

برنامه Dhrystone از ممیز شناور استفاده نمی‌کند. برنامه‌های معمولی نیز همین طور ...

ریک ریچاردسون از برنامه ریزان 1988) Dhrystone)

"این برنامه نتیجه تحقیقات زیاد روی برنامه‌های معمولی فرترن می‌باشد. نتیجه اجرای این برنامه روی ماشین‌های مختلف نشان می‌دهد که کدام ماشین برنامه‌های فرترن را بهتر اجرا می‌کند. دستورات عمداً طوری نوشته شده‌اند که توسط کامپایلر بهینه نشوند."

کورنو و ویچ من "از توضیحات برنامه ارزیابی1976) Whetstone)

یک کاربر کامپیوتر که برنامه خاصی را هر روزه اجرا می‌کند کاندید مناسبی برای ارزیابی یک کامپیوتر جدید می‌باشد. برای ارزیابی یک سیستم جدید، کاربر به سادگی می‌تواند زمان اجرای کامل کار یعنی ترکیب برنامه و دستورات سیستم عامل را که کاربر روی ماشین اجرا می‌کند، را اندازه بگیرد. ولی این روش عمومی‌نمی‌باشد. اکثراً سعی در یافتن روش‌های دیگری برای ارزیابی یک ماشین جدید دارند تا بتوانند کارآیی یک ماشین جدید را پیش بینی کنند. پنج سطح از برنامه‌ها برای این کار وجود دارد که به ترتیب کم شدن دقت پیش بینی، در اینجا ذکر می‌شوند:

۱ – برنامه‌های کاربردی واقعی، گرچه یک خریدار نمی‌داند که چند درصد از وقت کامپیوتر صرف این برنامه‌ها خواهدشد، ولی او می‌داند که بعضی از کاربرها از این کامپیوترها برای اجرای این برنامه‌ها برای حل مسائل واقعی استفاده خواهند کرد. مثالی از این برنامه‌ها کامپایرهای C ، ویراستارهای متن مانند Word و کاربردهای دیگری مانند فتوشاپ می‌باشند. برنامه‌های واقعی ورودی، خروجی و گزینه‌های انتخاب دارند که کاربر در زمان اجرای برنامه آنها را انتخاب می‌کند. یکی از معایب اصلی انتخاب این برنامه‌ها بعنوان برنامه ارزیابی این است که این برنامه‌ها معمولاً وابسته به ماشین و سیستم عامل

هستند و برای اجرای آنها روی یک کامپیوتر جدید معمولاً باید برنامه‌ها را تغییر داده و کارهای اساسی نظیر تغییر واسط گرافیکی که وابسته به ماشین می‌باشد.، را انجام داد.

۲ – برنامه‌های کاربردی تغییر بافته – در بسیاری از مواقع برنامه‌های واقعی را بعنوان سنگ بنای ساختن یک برنامه ارزیابی استفاده می‌شوند. این برنامه تغییر داده یا شبیه سازی می‌شوند. برنامه‌ها به دو دلیل اصل تغییر داده می‌شوند یکی برای سادگی حمل و دیگری برای ارزیابی کارآیی خاصی از ماشین، مثلاً برای یک برنامه ارزیابی کارآیی پردازنده، قسمت‌های ورودی و خروجی حذف شده یا به حداقل رسانده می‌شوند تا اثر آنها روی زمان اجرا حداقل شود. برنامه‌های شبیه سازی شده برای شبیه سازی برنامه‌هایی که بصورت محاوره ای هستند مانند شبیه سازی یک سیستم محاوره ای چند کاربره مانند یک سرویس دهنده، می‌باشند.

۳ – هستهٔ برنامه‌ها (0– Kernel) کوشش هایی برای ساخت یا استخراج قسمتهای کلیدی برنامه‌های واقعی و استفاده از آنها برای ارزیابی کارآیی شده است. دو مثال مشهور "Livemore Loop" و "Linpack" هستند. بر خلاف برنامه‌های واقعی، هیچ کاربری این هسته برنامه‌ها را اجرا نمی‌کند و این برنامه‌ها فقط برای ارزیابی مناسب هستند. هستهٔ برنامه‌ها روش مناسبی برای مجزا کردن ویژگی‌های مختلف کامپیوترها و بیان دلایل اختلاف در کارآیی برنامه‌های واقعی می‌باشد.

۴ – برنامه‌های ارزیابی کوچک (اسباب بازی ) – برنامه‌های ارزیابی کوچک معمولاً بین ۱۰ تا ۱۰۰ خط از دستورات بوده و نتیجه اجرای آن از قبل مشخص می‌باشد. برنامه هایی مانند مرتب سازی سریع و غربال اراتستن، نمونه‌های معروف این نوع برنامه‌ها هستند . این برنامه‌ها کوچک بوده و تایپ انها ساده

بوده و روی اکثر کامپیوتر ها قابل اجرا می‌باشند. بهترین کاربرد این برنامه‌ها تکالیف اولیه برنامه نویسی می‌باشد.

۵ - برنامه‌های ارزیابی مصنوعی – مشابه آنچه در مورد هستهٔ برنامه‌ها بیان شد، برنامه‌های ارزیابی مصنوعی سعی دارند که متناسب با عملکرد ها و عملونه‌های مجموعه بزرگی از برنامه ها عمل کنند. Whetstone و Dhrystone از معروفترین برنامه‌های ارزیابی مصنوعی هستند. مشخصات و برخی از نکات ضعف این برنامه‌ها در بخش ۱-۹ بیان خواهد شد. هیچ کاربری در حالت عادی برنامه‌های ارزیابی مصنوعی را اجرا نمی‌کند چون این برنامه‌ها چیز خاصی که مورد نظر کاربر باشد را محاسبه نمی‌کنند. برنامه‌های هسته ای از برنامه‌های واقعی استخراج می‌شوند ولی برنامه‌های ارزیابی مصنوعی بصورت مصنوعی ساخته می‌شوند تا مانند برنامه‌ها معمولی اجرا شوند. برنامه‌های ارزیابی مصنوعی بر خلاف برنامه‌های هسته ای، قسمتی از یک برنامه واقعی نیستند. چون شرکتهای کامپیوتری با توجه به قیمت و کارایی محصولات خود در بازار موفق هستند پس منابع زیادی برای بالا بردن کارآیی برنامه‌هایی که برای ارزیابی استفاده می‌شوند وجود دارند. این فشار باعث می‌شود که مهندسین سخت افزار و نرم افزار سعی و تلاش کنند که سیستم ها را طوری بهینه کنند که برنامه‌های ارزیابی مصنوعی، برنامه‌های ارزیابی اسباب بازی، برنامه‌های هسته ای و حتی برنامه‌های واقعی را بهتر اجرا کنند. البته آخرین آنها یعنی اجرای بهتر برنامه‌های واقعی مشکل بوده ولی غیر ممکن نیست. این حقیقت باعث شده که ارائه کنندگان برنامه‌های ارزیابی مشخص کنند که تحت چه شرایطی و چه کامپایلرهایی عمل کنند این مسئله به زودی تشریح می‌شود.

## اجتماع برنامه‌های ارزیابی

امروزه مرسوم است که از چندین برنامه ارزیابی با همدیگر برای‌اندازه گیری کارآیی یک پردازنده در کاربردهای گوناگون استفاده کنند. البته این مجموعه ها به‌اندازهٔ هر کدام از برنامه‌های ارزیابی خوب هستند. ولی مزیت این مجموعه ها این است که اگر ضعفی در یک برنامه ارزیابی باشد توسط دیگر برنامه‌ها پوشانده می‌شود. این مزیت بخصوص اگر معیار ارزیابی کارآیی مجموع زمان صرف شده برای کل مجموعه باشد، مشخص تر می‌باشد همچنین ممکن است با این مسئله که سیستمی برای یک برنامه ارزیابی خاص بهینه شده باشد، مبارزه شود. در این بخش نقاط قوت و ضعف روشهای مختلف جمع‌بندی کارآیی بحث می‌شود.

یکی از تلاشهای موفق برای خلق یک مجموعهٔ موفق از برنامه‌های ارزیابی برنامه SPEC مخف Standard Performance Evaluation Corporation است که در سال ۱۹۸۰ عرضه شد و برای ارزیابی بهتر ایستگاه‌های کاری ساخته شده بود. همانطور که صنعت کامپیوتر با گذشت زمان پیشرفت می‌کند، نیاز به برنامه‌های ارزیابی جدید نیز احساس می‌شود و امروزه برنامه‌های SPEC برای پوشش کاربردهای مختلف مبتنی بر مدل اولیه SPEC موجود می‌باشند. مستندات و گزارشهای مربوط به برنامه SPEC در سایت www.spec.org موجود می‌باشد. گرچه در بخش‌های آینده تکیه روی برنامه SPEC شده است ولی مجموعه زیادی از برنامه‌های ارزیابی برای کامپیوترهای شخصی تحت سیستم عامل ویندوز و برای محیط‌های کاربردی متفاوت موجود هستند و تعدادی از آنها در شکل ۱۱ نشان داده شده‌اند.

| Benchmark name | Benchmark description |
|---|---|
| Business Winstone | Runs a script consisting of Netscape Navigator and several office suite products (Microsoft, Corel, WordPerfect). The script simulates a user switching among and running different applications. |
| CC Winstone | Simulates multiple applications focused on content creation, such as Photoshop, Premiere, Navigator, and various audio-editing programs. |
| Winbench | Runs a variety of scripts that test CPU performance, video system performance, and disk performance using kernels focused on each subsystem. |

شکل ۱۱ – نمونه ای از برنامه‌های ارزیابی کامپیوترهای شخصی

دو برنامه اول مجموعه‌ای از برنامه‌های واقعی هستند ولی آخری ترکیبی از برنامه‌های هسته‌ای و برنامه‌های ارزیابی مصنوعی می‌باشد. این برنامه‌ها توسط انتشارات Ziff Davis که دربارهٔ کامپیوترهای شخصی مطلب منتشر می‌کند، مدیریت می‌شود. همچنین Ziff Davis آزمایش‌های مستقلی را انجام می‌دهد. برای اطلاعات بیشتر در مورد این برنامه‌های ارزیابی به سایت /www.etestinglabs.com/benchmarks مراجعه کنید.

## برنامه‌های ارزیابی کامپیوترهای رومیزی(Desktop Benchmark)

برنامه‌های ارزیابی کامپیوترهای رومیزی، به دو دستهٔ عمده تقسیم می‌شوند: برنامه‌های ارزیابی پردازنده و برنامه‌های ارزیابی گرافیک( گرچه برنامه‌های ارزیابی گرافیک در عین حال پردازنده را نیز بررسی می‌کنند.) برنامه SPEC در ابتدا برای ارزیابی پردازنده ساخته شد( در ابتدا SPEC89 نامیده شد) و تا اکنون ۴ نسل داشته است : SPEC CPU2000 که بعد از SPEC95 و SPEC92 به بازارآمد.(شکل ۱-۳۰ در بخش۱-۹ توسعه برنامه‌های ارزیابی را نشان می‌دهد.) برنامهSPEC CPU2000، خلاصه شده در شکل ۱۲، از ۱۱ برنامه ارزیابی اعداد صحیح (CINT2000) و ۱۴ برنامه ارزیابی ممیز شناور (CFP2000) تشکیل شده است. برنامه ارزیابی SPEC برنامه‌های واقعی هستند که جهت قابل حمل بودن تغییر داده شده‌اند و ورودی و خروجی آن حداقل شده است. ارزیابی اعداد صحیح شامل قسمتی از کامپایلر C، یک برنامه مسیریابی VLSI و یک کاربرد گرافیکی می‌باشد. ارزیابی ممیز شناور شامل کدهای مربوط به کوانتوم، مدل‌های اجزای محدود و دینامیک مایعات می‌باشد. مجموعهٔ

SPEC CPUبرای ارزیابی پردازنده کامپیوترهای رومیزی و سرویس دهنده‌های تک پردازنده می‌باشد. اطلاعات زیادی از این برنامه‌ها در این کتاب خواهد آمد. در قسمت بعدی بیان می‌شود که SPEC 2000 چگونه مشخصات ماشین، کامپایلر و سیستم عامل را گزارش می‌دهد. در بخش ۱-۹ بعضی از مشکلاتی که در هنگام توسعه مجموعه SPEC رخ می‌دهد و همچنین مسائل مربوط به نگهداری یک برنامه مفید ارزیابی بیان می‌شود.

گرچه SPEC CPU2000 برای ارزیابی کارآیی پردازنده می‌باشد ولی دو نوع برنامه ارزیابی گرافیک توسط SPEC تولید شده است: (سایت www.spec.org را ملاحظه کنید) یکی SPECviewperf که برای ارزیابی سیستم گرافیکی که از openGL پشتیبانی می‌کنند طراحی شده و دیگری SPECapc برای کاربردهایی که گرافیک زیادی دارند، SPECviewperf کارآیی چرخش یک تصویر سه بعدی را در سیستمی‌که تحت openGL باشد را بررسی می‌کند و یک مدل گرافیکی دارد و تعدادی از توابع openGL را فراخوانی می‌کند. SPECapc از اجزای چندین برنامه کاربردی شامل برنامه‌های زیر تشکیل می‌شوند:

برنامه pro/Engineer – یک برنامه مدل سازی مسطح که تعداد زیادی عملیات ۳ بعدی دارد. ورودی این برنامه مدل یک ماشین فتوکپی است که از ۳۷۰۰۰۰ قطعه تشکیل شده است.

برنامه Solidworks2001 یک ابزار طراحی سه بعدی CAD/CAM که ۵ آزمایش مختلف و متفاوت از آزمایش مبتنی بر I/O تا آزمایش مبتنی بر پردازنده را اجرا می کند. بزرگترین ورودی آن یک خط مونتاژ شامل ۲۷۶۰۰۰ قطعه می‌باشد.

برنامه UnigraphicV15 – مبتنی بر مدل هواپیما که مونتاژ و نمونه سازی و کنترل و بهینه سازی را دربر می‌گیرد. ورودی آن تمام قطعات یک هواپیما می‌باشد.

| Benchmark | Type | Source | Description |
|---|---|---|---|
| gzip | Integer | C | Compression using the Lempel-Ziv algorithm |
| vpr | Integer | C | FPGA circuit placement and routing |
| gcc | Integer | C | Consists of the GNU C compiler generating optimized machine code |
| mcf | Integer | C | Combinatorial optimization of public transit scheduling |
| crafty | Integer | C | Chess-playing program |
| parser | Integer | C | Syntactic English language parser |
| eon | Integer | C++ | Graphics visualization using probabilistic ray tracing |
| perlmbk | Integer | C | Perl (an interpreted string-processing language) with four input scripts |
| gap | Integer | C | A group theory application package |
| vortex | Integer | C | An object-oriented database system |
| bzip2 | Integer | C | A block-sorting compression algorithm |
| twolf | Integer | C | Timberwolf: a simulated annealing algorithm for VLSI place and route |
| wupwise | FP | F77 | Lattice gauge theory model of quantum chromodynamics |
| swim | FP | F77 | Solves shallow water equations using finite difference equations |
| mgrid | FP | F77 | Multigrid solver over three-dimensional field |
| apply | FP | F77 | Parabolic and elliptic partial differential equation solver |
| mesa | FP | C | Three-dimensional graphics library |
| galgel | FP | F90 | Computational fluid dynamics |
| art | FP | C | Image recognition of a thermal image using neural networks |
| equake | FP | C | Simulation of seismic wave propagation |
| facerec | FP | C | Face recognition using wavelets and graph matching |
| ammp | FP | C | Molecular dynamics simulation of a protein in water |
| lucas | FP | F90 | Performs primality testing for Mersenne primes |
| fma3d | FP | F90 | Finite element modeling of crash simulation |
| sixtrack | FP | F77 | High-energy physics accelerator design simulation |
| apsi | FP | F77 | A meteorological simulation of pollution distribution |

شکل ۱۲ برنامه‌های موجود در مجموعهٔ ارزیابی SPEC CPU2000

شامل ۱۱ برنامه صحیح (که همگی به زبان C نوشته شده‌اند به جز یک برنامه که به زبان C++ می‌باشد.) که برای‌اندازه‌گیری CINT2000 می‌باشد و ۱۴ برنامه ممیز شناور (۶ برنامه به زبان فرترن ۷۷ و ۵ برنامه به زبان C و ۳ برنامه به زبان فرترن ۹۰) که برای‌اندازه‌گیری CFP2000 می‌باشند. برای مشاهده اطلاعات بیشتر در مورد این مجموعهٔ ارزیابی به سایت www.spec.org مراجعه کنید.

## برنامه‌های ارزیابی سرویس دهنده‌ها (Server Benchmarks)

چون یک سرویس دهنده باید چندین عمل مختلف انجام دهد پس به چندین ارزیابی نیاز دارد. ساده‌ترین ارزیابی، ارزیابی بازده پردازنده می‌باشد. SPEC CPU2000 از ارزیابی SPEC CPU برای

ارزیابی پردازنده استفاده می‌کند و در سیستم‌های چند پردانده به ازای هر پردازنده یک کپی از برنامه SPEC CPU اجرا می‌شود و نرخ زمان پردازنده مشخص می‌شود. به همین دلیل به این اندازه‌گیری SPECrate می‌گویند.

علاوه بر SPECrate، اکثر کاربردهای سرویس دهنده‌ها مقدار زیادی فعالیت ورودی و خروجی دارند از انتقالات دیسک در سرویس دهنده‌های فایل و نقل و انتقالات شبکه در سرویس دهنده‌های شبکه و سرویس دهنده‌های پایگاه داده وسیستم‌های پردازش مبادلات می‌باشد. SPEC یک برنامه برای ارزیابی سرویس دهنده فایل(SPECSFS) و یک برنامه ارزیابی سرویس دهنده شبکه (SPECWeb) دارد. SPECSFS یک برنامه ارزیابی برای‌اندازه‌گیری کارآیی NSF مخففNetwork File System می‌باشد که از تقاضاهای فایل تشکیل می‌شود، این برنامه آزمایش کارآیی I/O (ورودی خروجی دیسک شبکه) را انجام می‌دهد، همچنین کارآیی پردازنده را نیز نشان می‌دهد. SPECSFS یک برنامه ارزیابی مبتنی بر کارآیی بوده ولی زمان پاسخگویی نیز در آن مهم می‌باشد. SPECweb یک برنامه ارزیابی سرویس دهنده شبکه بوده و تقاضای چندین متقاضی که تقاضای صفحات ایستا و پویا می‌کنند یا داده‌ای را برای سرویس دهنده می‌فرستند، را شبیه سازی می‌کند.

برنامه‌های ارزیابی پردازش تبادلات یا TP (مخفف Transaction Processing ) قابلیت یک سیستم برای انجام تبادلات رااندازه‌گیری می‌کند، این تبادلات شامل دسترسی به پایگاه داده و به روز رسانی آنها می‌باشد. رزرو بلیط هواپیما یا یک سیستم عابر بانک (ATM) نمونه‌های ساده یک سیستم پردازش تبادلات هستند، سیستم‌های پیچیده‌تر شامل پایگاه‌های دادهٔ پیچیده و تصمیم‌گیری بر مبنای آنها است. در اواسط دههٔ ۱۹۸۰ تعدادی از مهندسین مستقل از شرکت‌ها یک مجتمع پردازش تبادلات (TPC)

مخفف Transaction processing council تشکیل داده و یک سری برنامه‌های ارزیابی واقعی و عادلانه برای پردازش تبادلات ایجاد کردند. TPC-A در سال ۱۹۸۵ عرضه شده و تا کنون با ۴ نمونه جدید جایگزین شده است. TPC-C در سال ۱۹۹۲ ایجاد شده و یک محیط پرسش (query) پیچیده را شبیه سازی می‌کرد. TPC-H برای پشتیبانی مدل‌های تصمیم گیری ad hoc که در آنها پرسش‌های جدید ربطی به پرسشهای قبل نداشته و از دانش پرسش‌های قبلی نمی‌توان پرسشهای بعدی را بهینه کرد و در نتیجه در آنها زمان پاسخگویی به سؤالات خیلی طولانی می‌باشد، TPC-R یک سیستم پشتیبانی تجاری زمانی که کاربر پرسشهای استاندارد را انجام می‌دهد، را شبیه سازی می‌کند. در TPC-R اطلاعات مربوط به پرسشهای قبلی نگهداری شده و سیستم DBMS می‌تواند برای پرسشهای بعدی بهینه شود. TPC-W یک برنامه ارزیابی تبادلات مبتنی بر شبکه می‌باشد که تبادلات مبتنی بر تجارت را شبیه سازی می‌کند. این برنامه هم سیستم پایگاه داده را و هم نرم افزارهای سرویس دهنده شبکه را امتحان می‌کند. برنامه‌های ارزیابی TPC در سایت www.tpc.oig تشریح شده است.

تمام برنامه‌های ارزیابی TPC میزان تبادلات در ثانیه را اندازه‌گیری می‌کنند. همچنین آنها زمان پاسخگویی را اندازه گیری می‌کنند و کارآیی زمانی اندازه گیری می‌شود که زمان پاسخگویی در حد مشخص شده باشد. برای مدل کردن سیستم‌های جهان واقعی، باید نرخ تبادلات بالاتری برای سیستم‌های بزرگ در نظر گرفته شود، چه از نظر کاربران و چه از نظر پایگاه داده‌ای که این تبادلات مربوط به آن می‌باشد. در نهایت، باید هزینه سیستم نیز توسط برنامه ارزیابی کارآیی نیز در نظر گرفته شود تا مقایسهٔ درستی از نسبت هزینه – کارآیی بدست آید.

## ارزیابی سیستم‌های جاسازی شده (Embedded Benchmarks)

ارزیابی سیستم‌های جاسازی شده در مقایسه با ارزیابی سیستم‌های رومیزی یا سرویس دهنده‌ها پدیدهٔ نوظهوری می‌باشد. در حقیقت تعدادی از تولید کنندگان از برنامه ارزیابی Dhrystone استفاده می‌کنند، برنامه‌ای که بیش از ۱۰ سال است که از رده خارج شده است . همانطور که قبلاً بیان شد، سیستم‌های جاسازی شده کاربردها و امکانات متفاوتی دارند (بلادرنگ، تقریباً بلادرنگ و سیستم‌های موازنهٔ کارآیی و هزینه) بنابراین استفاده از یک برنامه ارزیابی خاص غیر واقعی می‌باشد. در عمل اکثر سازندگان سیستم‌های جاسازی شده برای ارزیابی یا از هستهٔ برنامه‌های کاربردی یا از نمونهٔ مستقل کاربرد اصلی استفاده می‌کنند.

برای کاربردهایی که می‌توان کارآیی آن را با هستهٔ آن بیان کرد، بهترین ارزیاب استاندارد EEMBC که. مخفف EDN Embedded Microprocessor Benchmark Consortium بوده و embassy تلفظ می‌شود.ارزیاب EEMBC از پنج کلاس تشکیل می‌شود: صنعتی یا اتوماسیون، مصرفی، شبکه، اتوماسیون دفتری و مخابرات.

شکل ۱۳ پنج کلاس مختلف را که شامل ۳۴ ارزیابی می‌باشد، نشان می‌دهد.

| Benchmark type | Number of kernels | Example benchmarks |
|---|---|---|
| Automotive/industrial | 16 | 6 microbenchmarks (arithmetic operations, pointer chasing, memory performance, matrix arithmetic, table lookup, bit manipulation), 5 automobile control benchmarks, and 5 filter or FFT benchmarks |
| Consumer | 5 | 5 multimedia benchmarks (JPEG compress/decompress, filtering, and RGB conversions) |
| Networking | 3 | Shortest-path calculation, IP routing, and packet flow operations |
| Office automation | 4 | Graphics and text benchmarks (Bézier curve calculation, dithering, image rotation, text processing) |
| Telecommunications | 6 | Filtering and DSP benchmarks (autocorrelation, FFT, decoder, encoder) |

شکل ۱۳- مجموعه ارزیابی EEMBC

این مجموعه شامل ۳۴ هسته از پنج کلاس مختلف است. برای اطلاعات بیشتر و امتیازات آن به سایت www.eembc.org مراجعه کنید.

## گزارش نتایج کارآیی

راهنمایی کلی برای گزارش اندازه گیری کارآیی، تکرار پذیری (reproducibility) می‌باشد یعنی یک محقق دیگر بتواند آن نتایج را دوباره بدست آورد. گزارش برنامه ارزیابی SPEC نیاز دارد که شرح کاملی از ماشین و پرچم‌های کامپایلر داده شده و نتایج اولیه و بهینه شده چاپ شود. بطور مثال، شکل ۱۴ قسمتی از گزارش SPEC CINT2000 در مورد ایستگاه کاری Dell Precision4/0 را نشان می‌دهد.

علاوه بر سخت افزار، نرم افزار و پارامترهای تنظیم شده، یک گزارش SPEC شامل زمانهای کارآیی واقعی بصورت جدول و بصورت گراف می‌باشد. یک گزارش ارزیابی TPC پیچیده‌تر می‌باشد چون علاوه بر نتایج ارزیابی‌ها شامل اطلاعات هزینه‌ها نیز می‌باشد.

نحوه سازماندهی نرم افزارهای یک سیستم می‌تواند تا حد زیادی روی نتایج ارزیابی اثر بگذارد. بطور مثال، کارآیی و پشتیبانی سیستم عامل می‌تواند در ارزیابی یک سرویس دهنده مهم باشد. به همین دلیل برنامه‌های ارزیابی را در حالت تک کاربره اجرا می‌کنندتا سربارها کاهش یابد. بعلاوه بعضی مواقع سیستم عامل تنظیم می‌شود تا نتیجه یک ارزیابی TPC بهتر گزارش شود. به همین صورت فناوری کامپایلر می‌تواند اثر زیادی بخصوص اگر کامپایلر اجازهٔ تغییر برنامه ورودی را داشته باشد (مثال مربوط به ارزیابی EEMBC در شکل ۱-۳۰ از بخش ۱-۹ را ملاحظه کنید) یا زمانی که یک ارزیابی به بهینه شدن حساس باشد (مثال توضیح داهد شده در مورد SPEC در صفحه را ملاحظه کنید.) داشته باشد. به همن دلایل مهم است که دقیقاً سیستم نرم افزاری اندازه گیری شده و اگر تغییرات غیر استانداردی انجام شده، مشخص شود. روش دیگر برای تنظیم نرم‌افزار برای بهبود نتیجه ارزیابی استفاده از پرچم‌های اختصاصی ارزیابی می‌باشد، این پرچم‌ها باعث تبدیلاتی می‌شوند که در اکثر برنامه‌ها غیر مجاز بود. یا باعث کاهش کارآیی در برنامه‌های دیگر می‌شوند. برای محدود کردن این فرآیند و افزایش دقت نتایج

SPEC، سازمان دهندگان SPEC اندازه گیری شرایط اولیه را علاوه براندازه گیری کارآیی بهینه شده انجام می‌دهند. شرایط اولیه باعث محدود شدن تولید گنندگان به استفاده از یک کامپایلر خاص و مجموعه مشخصی از پرچم‌ها برای تمام برنامه‌های مربوط به یک زبان (C یا فرترن) می‌باشند. شکل ۱۴ پارامترهای مربوط به شرایط اولیه را نشان می‌دهد، در بخش ۱-۹ پارامترهای قابل تنظیم برای بهینه کردن کارآیی روی این ماشین‌ها بیان خواهد شد.

علاوه بر سوالات مربوط به پرچم‌ها و بهینه سازی سؤال دیگر این است که آیا تغییر کد برنامه امکان پذیر است یا خیر و اینکه آیا استفاده از برنامه‌های اسمبلی نوشته شده با دست مجاز هست یا خیر. چهار نظر متفاوت در مورد این سؤالات مطرح است:

۱- تغییر کد برنامه‌ها مجاز نباشد. ارزیابی SPEC از این دسته بوده و اکثر برنامه‌های ارزیابی کامپیوترهای شخصی از این نوع هستند.

۲- تغییر کد برنامه‌ها مجاز ولی غیرممکن یا مشکل می‌باشد. برنامه‌های ارزیابی مانند TPC-C روی پایگاه‌های داده استاندارد نظیر اوراکل یا سرویس دهنده Sql مربوط به مایکرو سافت می‌باشند. گرچه این شرکت‌ها مایل هستند که ارزیابی بالایی داشته باشند ولی حاضر نیستند برنامه‌های خود را طوری تغییر دهند که نتیجه ارزیابی یک مشتری خاص بهتر شود. همچنین TPC-C مقدار زیادی به سیستم عامل وابسته می‌باشد سیستم عامل قابل تغییر می‌باشد ولی تغییر در آن نیازمند ویرایش بعدی سیستم عامل خواهد بود.

۳- تغییر کد برنامه مجاز می‌باشد. مجموعه‌های ارزیابی سوپر کامپیوترها اجازهٔ تغییر کد برنامه‌ها را دارند. بطور مثال برنامه ارزیابی NAS که برای سوپر کامیوترها می‌باشد ورودی و خروجی را مشخص

کرده و کد ارائه شده توسط تولیدکنندگان قابل تغییر می‌باشد، حتی تولید کنندگان می‌توانند الگوریتم‌ها را نیز عوض کنند ولی نباید خروجی برنامه تغییر کند. ارزیاب EEMBC اجازهٔ تغییر کد برنامه را می‌دهد و نتایج بدست آمده در گزارش‌ها "نتایج بهینه شده" ولی نتایج بدون تغییر در کدها "نتایج واقعی" نامیده می‌شوند.

۴- نوشتن دستی کدها مجاز می‌باشد. EEMBC تغییر کدهای اسمبلی برنامه ارزیاب را مجاز می‌داند. هستهٔ کوچک آن این کار را امکان پذیر می‌کند ولی برای کاربردهای جاسازی شده بزرگ به جز حلقه‌های کوچک، این کار قابل استفاده نیست.

| Hardware | | Software | |
|---|---|---|---|
| Model number | Precision WorkStation 410 | O/S and version | Windows NT 4.0 |
| CPU | 700 MHz, Pentium III | Compilers and version | Intel C/C++ Compiler 4.5 |
| Number of CPUs | 1 | Other software | See below |
| Primary cache | 16KBI+16KBD on chip | File system type | NTFS |
| Secondary cache | 256KB(I+D) on chip | System state | Default |
| Other cache | None | | |
| Memory | 256 MB ECC PC100 SDRAM | | |
| Disk subsystem | SCSI | | |
| Other hardware | None | | |

**SPEC CINT2000 base tuning parameters/notes/summary of changes:**

+FDO: PASS1=-Qprof_gen PASS2=-Qprof_use

Base tuning: -QxK -Qipo_wp shlW32M.lib +FDO

shlW32M.lib is the SmartHeap library V5.0 from MicroQuill www.microquill.com

Portability flags:

176.gcc: -Dalloca=_alloca /F10000000 -Op

186.crafy: -DNT_i386

253.perlbmk: -DSPEC_CPU2000_NTOS -DPERLDLL /MT

254.gap: -DSYS_HAS_CALLOC_PROTO -DSYS_HAS_MALLOC_PROTO

شکل ۱۴ - گزارش پایهٔ CINT2000 دربارهٔ ماشین، نرم افزار و پارامترهای پایه برای Dell Precision4/0 این اطلاعات مربوط به گزارش پایهٔ CINT2000 می‌باشد. این داده در سایت www.spec.org/osg/cpu2000/results/cpu2000.html در دسترس می‌باشد.

نکته کلیدی که طراحان ارزیابی باید برای مجاز کردن تغییر کدها در نظر بگیرند این است که آیا چنین تغییری عملیات واقعی را منعکس می‌کند و برای کاربر مفید است یا اینکه فقط برای کاهش دقت ارزیابی در پیش بینی کارآیی واقعی می‌باشد.

## مقایسه و جمع بندی کارآیی

مقایسه کارآیی کامپیوترها بندرت کسالت آور می‌باشد، بخصوص اگر طراحان کامپیوتر ذینفع باشند. اتهام زدن و متهم کردن دیگران در اینترنت مرسوم است، یکی به تاکتیک‌های پنهانی و دیگری به استفاده از جملات گمراه کننده متهم می‌شود چون ادامهٔ حیات بعضی از سیستم‌ها بستگی به نتایج این مقایسه‌های کارآیی دارد پس قابل درک است که حقیقت به ندرت بیان شود. ولی اخیراً می‌توان تفاوتها را با فرض‌های مختلف یا کمبود اطلاعات بیان کرد.

می‌توان تصور کرد که اگر افراد در برنامه‌ها، محیط آزمایش و تعریف سریعتر توافق داشتند، اختلافات به پایان می‌رسید و شبکه برای کارهای تحقیقهای آزاد می‌شد. ولی متاسفانه این طور نیست. حتی اگر روی مبناها توافق شود، بر سر اینکه کارآیی چندین برنامه چگونه جمع بندی می‌شوند اختلاف رخ می‌دهد. مثلاً دو مقاله مختلف در یک جمله ممکن است روشهای متفاوتی برای جمع بندی ارائه کنند. شکل ۱۵ از یکی از این مقالات گرفته شده و مثالی از این ابهامات می‌باشد.

| | Computer A | Computer B | Computer C |
|---|---|---|---|
| Program P1 (secs) | 1 | 10 | 20 |
| Program P2 (secs) | 1000 | 100 | 20 |
| Total time (secs) | 1001 | 110 | 40 |

شکل ۱۵ زمان اجرای دو برنامه روی سه کامپیوتر مختلف

داده‌ها از شکل شماره یک مقاله اسمیت در سال ۱۹۸۸ گرفته شده است.

## اندازه گیری زمان اجرای کلی

ساده‌ترین روش برای جمع بندی نسبت کارآیی برای اجرای دو برنامه ، جمع کردن زمانهای آنها می‌باشد. معدل زمان اجرای چندین برنامه مختلف میانگین ریاضی آنها می‌باشد :

که: $Time_i$ زمان اجرای برنامه I ام از n کار مختلف می‌باشد.

## وزن دهی زمان اجرا Weighted Execution Time

سؤالی که باقی می‌ماند این است که ترکیب مناسب برنامه‌ها چگونه می‌باشد؟ آیا برنامه‌های p1 و p2 بطور یکسان اجرا می‌شوند و فرض قبل برای محاسبهٔ میانگین درست است یا خیر؟ اگر احتمال آنها یکسان نباشد با دوروش مختلف می‌توان کارآیی مجموع را محاسبه کرد. روش اول این است که به هر برنامه یک ضریب وزن دهی: W نسبت دهیم که این وزن بستگی به نرخ تکرار آن برنامه در بار کاری دارد. مثلاً اگر۲٪ برنامه‌های موجود در بارکاری برنامه p1 و ۸۰٪ از برنامه‌ها p2 باشند آنگاه ضریب وزن دهی ۲٪ و ۸٪ می‌باشد(مجموع تمام وزن‌ها باید ۱ شود.) با محاسبه حاصل جمع زمان اجرای هر برنامه ضربدر وزن آن، تصویر روشنی از بارکاری بدست می‌آید. این مجموع، میانگین حسابی وزن‌دار نامیده می‌شود. شکل ۱۶ داده‌های شکل ۱۵ را با ۳ وزن مختلف نشان می‌دهد، که هرکدام متناسب با زمان اجرا در بارکاری ترکیبی می‌باشد.

| | Programs | | | Weightings | | |
|---|---|---|---|---|---|---|
| | A | B | C | W(1) | W(2) | W(3) |
| Program P1 (secs) | 1.00 | 10.00 | 20.00 | 0.50 | 0.909 | 0.999 |
| Program P2 (secs) | 1000.00 | 100.00 | 20.00 | 0.50 | 0.091 | 0.001 |
| Arithmetic mean: W(1) | 500.50 | 55.00 | 20.00 | | | |
| Arithmetic mean: W(2) | 91.91 | 18.19 | 20.00 | | | |
| Arithmetic mean: W(3) | 2.00 | 10.09 | 20.00 | | | |

شکل ۱۶ میانگین حسابی وزن‌دار و اجرای دو برنامه (p1 و p2) روی ۳ ماشین (A.B.C) با استفاده از ۳ وزن مختلف (W1.W2.W3)

## زمان اجرای نرمالیزه شده و مزایا و محاسن میانگین هندسی

روش دوم در مورد ترکیب نامتعادل برنامه‌ها در بارکاری این است که زمان اجرا نسبت به یک ماشین مبنا نرمالیزه شده و سپس میانگین زمان اجرای نرمالیزه شده محاسبه شود. این روش توسط برنامه ارزیابی SPEC استفاده می‌شود و زمان پایه روی یک ایستگاه SPARC بعنوان مبنا می‌باشد. این‌اندازه گیری احساس رضایت بخشی دارد چون کارآیی برنامه‌های جدید را می‌توان با ضرب کردن یک عدد در کارآیی آن در ماشین پایه بدست آورد.

میانگین زمان اجرای نرمالیزه شده را می‌توان بصورت میانگین حسابی یا میانگین هندسی بدست آورد. بر خلاف میانگین حسابی، میانگین هندسی زمان اجرای نرمالیزه شده از اینکه کدام ماشین مبنا انتخاب شود، مستقل می‌باشد. بنابراین میانگین حسابی نباید برای محاسبهٔ میانگین زمان اجرای نرمالیزه شده استفاده شود. شکل ۱۷ بعضی اختلافات استفاده از هر دو میانگین حسابی و هندسی در حالت نرمالیزه شده را نشان می‌دهد.

| | Normalized to A | | | Normalized to B | | | Normalized to C | | |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | A | B | C | A | B | C |
| Program P1 | 1.0 | 10.0 | 20.0 | 0.1 | 1.0 | 2.0 | 0.05 | 0.5 | 1.0 |
| Program P2 | 1.0 | 0.1 | 0.02 | 10.0 | 1.0 | 0.2 | 50.0 | 5.0 | 1.0 |
| Arithmetic mean | 1.0 | 5.05 | 10.01 | 5.05 | 1.0 | 1.1 | 25.03 | 2.75 | 1.0 |
| Geometric mean | 1.0 | 1.0 | 0.63 | 1.0 | 1.0 | 0.63 | 1.58 | 1.58 | 1.0 |
| Total time | 1.0 | 0.11 | 0.04 | 9.1 | 1.0 | 0.36 | 25.03 | 2.75 | 1.0 |

شکل ۱۷ – زمانهای اجرای نرمالیزه شده

شکل ۱۷ نرمالیزه نسبت به هرکدام از کامپیوترها – کارآیی میانگین حسابی وابسته به اینکه کدام ماشین مبنا باشد، متفاوت می‌باشد. در ستون ۲، زمان اجرای B، پنج برابر طولانی‌تر از A می‌باشد ولی در ستون ۴ این مساله برعکس می‌باشد. در ستون ۳ کامپیوتر C از همه آهسته‌تر می‌باشد ولی در ستون ۹ کامپیوتر C از همه سریعتر است. ولی میانگین هندسی مستقل از مبنای نرمالیزه شدن است، A و B کارآیی برابر و کامپیوتر C زمان اجرایی ۶۳/۰ برابر A و B دارد.(۱,۵۸/۱ نیز ۶۳/۰ است). متاسفانه مجموع زمان اجرای A ا زمان اجرای B، 10 برابر بیشتر بوده و زمان اجرای 3،B برابر زمان اجرای C می‌باشد. نکته جالب تناسب بین میانگین‌های هر مجموعه می‌باشد که همواره رابطه زیر برقرار است:

میانگین حسابی ≤ میانگین هندسی ≤ میانگین

چون وزنها در میانگین حسابی وزن‌دار وابسته به زمان اجرا یک کامپیوتر می‌باشد، همانندشکل ۱-۱۶، بنابراین نه تنها از میزان تکرار برنامه در بارکاری بلکه از کامپیوتر اختصاص داده شده واندازه ورودی برنامه نیز تأثیر می‌پذیرند. ولی میانگین هندسی زمان اجرای نرمالیزه شده، مستقل از زمان اجرای جداگانه یک برنامه بوده و اهمیتی ندارد که از چه کامپیوتری برای نرمالیزه کردن استفاده شود. حال اگراندازه گیری کارآیی برای حالتی باشد که در آن برنامه‌ها ثابت ولی ورودی برنامه متفاوت باشد، آنگاه مقایسه کننده می‌تواند نتایج نادرست از میانگین حسابی وزن‌دار بدست آورد، برای این کار بزرگترین ورودی را به برنامه ارزیابی که بهترین کارآیی را دارد اعمال می‌کند و باعث تغییر زمان اجرا می‌شود، در چنین موقعیتی میانگین هندسی کمتر از میانگین حسابی تأثیر می‌پذیرد.

بزرگترین عیب میانگین هندسی زمان اجرای نرمالیزه شده این است که اصلی اساسی‌اندازه گیری کارآیی یعنی پیش بینی زمان اجرا را نقص می‌کند. میانگین هندسی در شکل ۱-۱۷ بیان می‌کند که برای برنامه‌های p1 و p2،کارآیی کامپیوترهای A و B یکسان می‌باشد، ولی این مساله فقط وقتی درست می‌باشد که به ازای هر بار اجرای برنامه p2، برنامه p1 نیز ۱۰۰ بار اجرا می‌شود (شکل۱-۱۶) . زمان اجرای کلی برای این بارکاری بیان می‌کند که کامپیوترهای A و B هر کدام ۵۰٪ از کامپیوترهای C سریعتر هستند، در صورتی که میانگین هندسی می‌گویدکه C از A و B سریعتر می‌باشد. در حالت کلی، بارکاری که برای ۳ کامپیوتر یا بیشتر بتواند با کارآیی پیش بینی شده توسط میانگین هندسی زمان اجرای نرمالیزه شده تطبیق پیدا کند. دلیل اصلی بررسی میانگین هندسی برای کارآیی نرمالیزه شده این بود که از اینکه توجه یکسانی به برنامه‌های موجود در بارکاری شود اجتناب گردد، ولی آیا این کار پیشرفتی داشته است؟ عیب دیگر استفاده از میانگین هندسی بعنوان روشی برای جمع‌بندی کارآیی در یک مجموعهٔ

ارزیابی (مانند آنچه . SPEC PU2000 انجام می‌دهد) این است که طراحان سخت افزار و نرم افزار توجه خود را به ارزیابی که کارآیی آن را ساده‌تر می‌توان بهبود داد توجه کرده و به ارزیابی آهسته توجهی نمی‌کنند. مثلاً اگر بهبود سخت افزار یا نرم افزلر بتواند زمان اجرای یک ارزیابی را از ۲ ثانیه به ۱ ثانیه برساند، اثری که روی میانگین هندسی دارد مانند اثر پیشرفتی است که زمان یک ارزیابی ۱۰۰۰۰ ثانیه‌ای را به ۵۰۰۰ ثانیه تبدیل می‌کند.

ولی هر شخصی که این برنامه‌ها را اجراکند، کاردوم را کاری بزرگ و کار اول را بی استفاده می‌داند. معمولاً شکستن برنامه‌های کوچک ساده‌تر می‌باشد، یعنی کاری کرد که بدون بهبود کارآیی، نتایج ارزیابی بهتر شود و این مساله در مورد راه حال ایده‌آل‌اندازه گیری یک کار واقعی و وزن دهی هر برنامه با توجه به تکرار آن می‌باشد. اگر این کار امکان پذیر نباشد، نرمالیزه کردن که باعث می‌شود به هر برنامه زمان یکسانی روی یک کامپیوتر خاص اختصاص داده شود وزنها را مشخص‌تر کرده و پیش بینی زمان اجرای یک کار ترکیبی را مشخص می‌کند. مساله مشخص نبودن ورودی را نیز می‌توان با مشخص کردن ورودی در هنگام مقایسه کارآیی، حل کرد. اگر نتایج باید برای یک کامپیوتر خاص نرمالیزه شودند، ابتدا جمع بندی کارآیی را با توجه به وزن‌های مناسب‌اندازه گیری کرده و سپس نرمالیزه کردن را انجام دهید..

در انتها، باید به خاطر داشت که هر جمع بندی الزاماً باعث از دست رفتن اطلاعات می‌شود، بخصوص زمانی که‌اندازه گیری تغییرات زیادی داشته باشد. بنابراین مهم است که هم نتایج تک تک ارزیابی‌ها و هم جمع بندی در دسترس باشد. پس نتایج جمع بندی را باید با احتیاط استفاده کرد.، چون این جمع بندی ممکن است بیان کننده کارآیی یک کاربرد کاربر نباشد.

## ۱ –۶ اصول کمّی(Quantitative) در طراحی کامپیوتر

اکنون که تعریف، روش اندازه گیری و روش جمع بندی کارآیی گفته شد، می‌توان بعضی از اصول و راه کارهای مفید برای آنالیز و طراحی کامپیوتر را بیان کرد. در واقع، این بخش بعضی از ملاحظات مهم در طراحی با کارآیی بالا یا توازن هزینه و کارآیی و همچنین دو رابطه برای ارزش‌گذاری طراحی‌های مختلف را ذکر می‌کند.

### انجام سریع کارهای عمومی (Make the common case fast)

شاید مهم‌تری و مؤثرترین اصل در طراحی کامپیوتر انجام سریع کارهای عمومی‌می‌باشد: در هنگام تصمیم‌گیری بین طرحهای مختلف، باید اهمیت به حالات عمومی‌تر داده شود. این اصل همچنین در مشخص کردن نحوه مصرف منابع رعایت می‌شود چون سریع کردن بعضی از قسمت‌ها اگر آن قسمت احتمال وقوع بیشتری داشته باشد، مؤثرتر می‌باشد. بهبود وقایع مکرر به جای بهبود وقایع نادر به بالابردن کارآیی کمک می‌کند. به علاوه موارد پر استفاده معمولاً از موارد نادر ساده‌تر بوده و می‌توان آنها را سریعتر انجام داد. مثلاً وقتی که دو عدد را پردازنده با هم جمع می‌کند می‌توان وقوع سرریز را یک احتمال نادر در نظر گرفت و حالت بدون سر ریز را بعنوان یک حالت عمومی بهبود داد تا کارآیی بالاتر رود. این کار ممکن است باعث کند شدن حالتهایی که سر ریز رخ می‌دهد، بشود، ولی اگر سرریز بندرت رخ دهد، کارآیی کلی با بهینه سازی حالت معمولی بهبود می‌یابد.

تعداد زیادی از چنین حالتهایی در این کتاب بیان خواهد شد. برای اعمال این اصل ساده باید ابتدا مشخص کرد که حالات مکرر چه حالاتی هستند و با سریعتر کردن این حالات چقدر می‌توان کارآیی را افزایش داد. یک قانون اساسی که قانون اَمدال نامیده می‌شود، برای تأیید این اصل وجود دارد.

## قانون اَمدال Amdahl's law

میزان افزایش کارآیی که با بهبود قسمتی از کامپیوتر می‌توان بدست آورد با قانون امدال محاسبه می‌شود. قانون امدال بیان می‌کند که میزان افزایش کارآیی که می‌توان با افزایش زمان اجرای یک حالت کاری بدست آورد بستگی به میزان تکرار آن حالت کاری دارد.

قانون امدال میزان تسریع (Speedup) که می‌توان با استفاده از ویژگی خاصی بدست آورد را مشخص می‌کند. ولی تعریف تسریع چیست؟ فرض کنید که می‌توان بهبودی در کامپیوتر انجام داد که باعث افزایش کارآیی در هنگام استفاده شود. تسریع نسبت زیر خواهد بود:

$$\text{تسریع} = \frac{\text{کارآیی تمامی وظیفه‌ها زمانی که بهبود انجام شده}}{\text{کارآیی تمامی وظیفه‌ها بدون انجام بهبود}}$$

یا به عبارت دیگر

$$\text{تسریع} = \frac{\text{زمان‌اجرای تمامی وظیفه‌ها بدون انجام بهبود}}{\text{زمان‌اجرای تمامی وظیفه‌ها زمانی که بهبود انجام شده}}$$

تسریع مشخص می‌کند که انجام یک کار روی ماشین بهبود یافته در مقایسه با ماشین اولیه چقدر سریعتر اجرا می‌شود.

قانون امدال یک راه سریع برای یافتن تسریع برای یک بهبود می‌باشد که به دو فاکتور زیر وابسته می‌باشد:

۱ - درصدی از زمان محاسبه در کامپیوتر اولیه که می‌تواند از مزایای بهبود استفاده کند. مثلاً اگر ۲۰ ثانیه از برنامه‌ای که در ۶۰ ثانیه اجرا می‌شود، می‌تواند از بهبود سود ببرد، این درصد ۲۰/۶۰ خواهد

بود. این مقدار که درصد بهبود $Fraction_{enhanced}$ نامیده می‌شود، همواره کوچکتر یا مساوی یک می‌باشد.

۲- میزان پیشرفت در حالت اجرای بهبود یافته، یعنی اینکه یک کار اگر بهبود برای تمام برنامه باشد چقدر سریعتر انجام می‌شود. این مقدار نسبت زمان حالت اولیه به زمان بهبود یافته می‌باشد: مثلاً اگر قسمتی از کار که از بهبود استفاده می‌کند، ۲ ثانیه اجرا شود و همین قسمت در کامپیوتر اولیه در ۵ ثانیه اجرا شود، میزان بهبود ۵/۲ خواهد بود. این مقدار تسریع بهبود $Speedup_{enhanced}$ نامیده شده و همواره بزرگتر یا مساوی یک می‌باشد.

زمان اجرای یک کار کامل روی ماشین بهبود یافته مجموع زمان اجرای قسمت بهبود نیافته و زمان قسمت بهبود یافته می‌باشد:

$$\text{زمان‌اجرای جدید} = \text{زمان‌اجرای قبلی} \times \left( (1 - \text{درصد بهبود}) + \frac{\text{درصد بهبود}}{\text{میزان تسریع بهبود}} \right)$$

میزان تسریع کلی نسبت زمانهای اجرا خواهد بود:

$$\text{میزان تسریع کلی} = \frac{\text{زمان‌اجرای قبلی}}{\text{زمان‌اجرای جدید}} = \frac{1}{(1 - \text{درصد بهبود}) + \frac{\text{درصد بهبود}}{\text{میزان تسریع بهبود}}}$$

---

مثال: فرض کنید که پردازنده یک سرویس دهنده، که صفحات وب را سرویس می‌دهد، را بهبود دهیم. پردازنده جدید ۱۰ بار سریعتر از پردازنده قبلی صفحات وب را پردازش می‌کند. فرض کنید که پردازنده قبل ۴۰٪ مواقع محاسبات انجام داده و ۶۰٪ مواقع منتظر ورودی و خروجی باشد، میزان تسریع کلی بدست آمده با این بهبود را محاسبه کنید:

پاسخ:

درصد بهبود = ۰/۴

میزان تسریع بهبود = ۱۰

$$\text{میزان تسریع کلی} = \frac{1}{(1-0.4)+\frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$

---

قانون امدال بیان کننده قانون کاهش می‌باشد: یعنی میزان اثری که بالابردن میزان تسریع در بالابردن کارآیی دارد، هر چه بهبود بیشتر باشد اثر آن کمتر می‌شود. یک نتیجه مهم قانون امدال این است که اگر بهبود فقط برای قسمتی از یک کار قابل استفاده باشد نمی‌توان میزان تسریع را از یک منهای درصد آن قسمت بیشتر کرد.

یکی از خطاهای رایج در استفاده از قانون امدال، اشتباه شدن " درصد زمانی که از بهبود سود می‌برد" با "درصد زمان بعد از بهبود" می‌باشد. اگر به جای محاسبهٔ زمانی از کار که می‌تواند از بهبود سود ببرد، اندازه گیری زمان بعد از بهبود انجام شود، نتایج اشتباه خواهد بود! (مسئله ۱-۳ را برای پیدا کردن اشتباه ببینید.)

قانون امدال راهنمایی می‌کند که کارآیی را چقدر می‌توان بهبود داد و چگونه منابع را توزیع کنیم تا موازنهٔ هزینه و کارآیی برقرار شود. هدف به طور آشکار این است که نحوه صرف منابع متناسب با زمانی باشد که صرف می‌شود. قانون امدال برای مقایسه کارآیی کلی دو سیستم مختلف مناسب می‌باشد، همچنین می‌توان برای مقایسه دو پردازنده از این قانون استفاده کرد، همانطور که در مثال بعد آمده است.

---

مثال: یکی از تبدیلات پر استفاده در موتورهای گرافیکی ریشه دوم یا جذر می‌باشد. پیاده سازی جذر یک عدد ممیز شناور دارای کارآیی متفاوتی بخصوص در پردازنده‌های طراحی شده برای کارهای

گرافیکی، می‌باشد. فرض کنید جذر ممیز شناور ۲۰٪ از زمان اجرای یک برنامه ارزیابی گرافیکی باشد. یک راه حل برای بهبود کارآیی این باشد که سخت افزار مربوط به جذر را تغییر داد بطوریکه ۱۰ برابر سریعتر شود، راه حل دیگر تغییر کل واحد ممیز شناور می‌باشد بطوریکه تمام دستورات ممیز شناور ۱/۶ برابر سریعتر اجرا شود و دستورات ممیز شناور در کل ۵۰٪ از زمان یک کاربرد را تشکیل می‌دهند. برای تیم طراحی بهبود تمام دستورات ممیز شناور به میزان ۱/۶ برابر با بهبود قسمت محاسبهٔ جذر به میزان ۱۰ برابر یکسان می‌باشد. بهبود این دو روش را با هم مقایسه کنید.

پاسخ:

برای مقایسه این دو روش، میزان تسریع آنها را با هم مقایسه می‌کنیم:

$$\text{میزان تسریع جذر} = \frac{1}{(1-0.2)+\frac{0.2}{10}} = \frac{1}{0.82} \approx 1.22$$

$$\text{میزان تسریع تمام دستورات} = \frac{1}{(1-0.5)+\frac{0.5}{1.6}} = \frac{1}{0.8125} \approx 1.23$$

پس بهبود تمام دستورات ممیز شناور نتیجه بهتری دارد، چون میزان تکرار آنها بیشتر است.

---

در مثال فوق، نیاز است که زمان مصرف شده توسط واحد ممیز شناور بهبود یافته را اندازه بگیریم ولی معمولاً اندازه‌گیری مستقیم این زمان مشکل می‌باشد. در بخش بعدی، روش دیگری برای این مقایسه براساس یک معادله بیان می‌شود که در آن روش زمان اجرای پردازنده به سه جزء مجزا تقسیم می‌شود. اگر بتوان تأثیر یک تغییر را روی هر کدام از این اجزا بدست آورد می‌توان اثر تغییر روی کارآیی کلی را مشخص کرد. وانگهی شبیه‌سازهایی وجود دارند که می‌توان این اجزا را قبل از ساختن سخت افزار، بررسی کرد.

## فرمول کارآیی پردازنده (The CPU Performance Equation)

اصولاً تمام کامپیوترها از یک کلاک ساعت مرکزی استفاده می‌کنند که با نرخ ثابتی کار می‌کند. زمان ثابت کلاک را تیک تیک ساعت، پریود ساعت، سیکل ساعت یا سیکل می‌گویند. طراحان کامپیوتر کلاک ساعت را بر حسب مدت یک پریود (مثلاً یک نانو ثانیه ) یا بر حسب بسامد (مثلاً یک گیگا هرتز) بیان می‌کنند. زمان پردازنده برای یک برنامه را می‌توان به دو روش محاسبه کرد:

زمان یک کلاک * تعداد کلاک مورد نیاز برای برنامه = زمان پردازنده

یا

فرکانس کلاک / تعداد کلاک مورد نیاز برای برنامه = زمان پردازنده

علاوه بر شمارش تعداد کلاک مورد نیاز برای اجرای یک برنامه، می‌توان تعداد دستورالعمل‌هایی که اجرا می‌شوند و طول برنامه یا تعداد دستورالعمل (IC) مخفف Instruction Count نامیده می‌شوند را شمارش کرد. اگر تعداد دستورالعمل ها و تعداد کلاک مورد نیاز هر دستورالعمل در دست باشد می‌توان میزان متوسط کلاک برای هر دستورالعمل یا CPI مخفف Clock cycle Per Instruction را محاسبه کرد. چون کار کردن با آن ساده‌تر می‌باشد و در این فصل با پردازنده‌های ساده سر و کار داریم، از CPI استفاده می‌کنیم. بعضی از طراحان از تعداد دستورالعمل در هر کلاک یا IPC استفاده می‌کنند که معکوس CPI می‌باشد.CPI بصورت زیر محاسبه می‌شود:

$$CPI = \frac{\text{تعداد کلاک براي برنامه}}{\text{تعداد دستورالعمل‌ها}}$$

CPI معیار مناسبی برای انواع مختلف دستورالعمل‌ها و پیاده سازی‌های متفاوت بوده و بطور گسترده‌ای در این کتاب استفاده خواهد شد.

با منتقل کردن تعداد دستورالعمل‌ها به طرف چپ رابطه فوق تعداد کلاک برنامه برابر* CPI IC خواهد بود . پس می‌توان CPI را در فرمول مربوط به زمان اجرا بصورت زیر استفاده کرد:

تعداد دستورالعمل‌ها * تعداد کلاک هر دستور * زمان یک کلاک = زمان پردازنده

یا

فرکانس کلاک / (تعداد دستورالعمل‌ها * تعداد کلاک هر دستور) = زمان پردازنده

با گسترش فرمول اول بر اساس واحدهای اندازه‌گیری و معکوس کردن فرکانس کلاک نحوهٔ ارتباط این اجزا مشخص می‌شود:

$$\text{زمان پردازنده} = \frac{\text{ثانیه}}{\text{برنامه}} = \frac{\text{ثانیه}}{\text{کلاک}} \times \frac{\text{تعداد کلاک}}{\text{دستورالعمل}} \times \frac{\text{تعداد دستورالعمل‌ها}}{\text{برنامه}}$$

همانطور که این فرمول نشان می‌دهد، کارآیی پردازنده به سه عامل سیکل کلاک، (یا فرکانس آن)، تعداد سیکل هر دستورالعمل و تعداد دستورالعمل بستگی دارد. پس زمان پردازنده بطور یکسانی به این سه عامل بستگی دارد، مثلا بهبود ۱۰٪ درهر کدام از این عوامل باعث بهبود ۱۰٪ در زمان پردازنده خواهد شد.

متاسفانه امکان تغییر یک پارامتر مستقل از پارامترهای دیگر وجود ندارد چون فناوری که باعث تغییر یک پارامتر می‌شود از دیگر پارامترها مستقل نبوده و هر کدام به صورت زیر است:

کلاک سیستم–وابسته به فناوری سخت افزار و سازمان کامپیوتر

CPI – وابسته به سازمان کامپیوتر و معماری مجموعه دستورالعمل‌ها

تعداد دستورالعمل‌ها– وابسته به معماری مجموعه دستورالعمل‌ها و فناوری کامپایلر

خوشبختانه، اکثر تکنیک‌های بهبود کارآیی معمولاً یک پارامتر پردازنده را بهبود داده و اثر منفی کمی روی دیگر پارامترها دارند.

بعضی مواقع در طراحی پردازنده تعداد کلاک پردازنده برای اجرای برنامه‌ها محاسبه می‌شود:

$$\text{تعداد کلاک پردازنده} = \sum_{i=1}^{n} IC_i \times CPI_i$$

که: $IC_i$تعداد تکرار دستورالعمل I ام در یک برنامه و $CPI_i$ تعداد متوسط کلاک برای اجرای دستورالعمل I می‌باشد.

باتوجه به فرمول فوق می‌توان زمان پردازنده برای اجرای برنامه را بصورت زیر محاسبه کرد:

$$\text{زمان پردازنده} = \left( \sum_{i=1}^{n} IC_i \times CPI_i \right) \times \text{کلاک یک زمان}$$

در روش آخر برای محاسبه CPI، برای هر دستورالعمل $CPI_i$ بطور جداگانه و درصد وقوع آن دستورالعمل در برنامه (یعنی$IC_I$ تقسیم بر تعداد دستورالعمل‌ها محاسبه می‌شود. $CPI_i$ باید واقعاً اندازه‌گیری شود و نباید از روی جداول راهنمای مرجع کامپیوتر محاسبه شود چون $CPI_i$ باید شامل اثر خط لوله، باخت‌های حافظه نهان و دیگر آثار غیر مطلوب سیستم حافظه باشد.

مثال قبل را در نظر بگیرید، در اینجا این مثال تغییر داده می‌شود تا میزان تکرار دستورالعمل و مقدار CPI دستورالعمل‌ها اندازه‌گیری شود، این کار در عمل با شبیه سازی یا وسایل‌اندازه‌گیری سخت افزاری انجام می‌شود.

---

مثال: فرض کنید که‌اندازه‌گیری‌های زیر انجام شده باشد:

میزان وقوع دستورات ممیز شناور(به جز دستور جذر) =۲۵٪

متوسط CPI برای عملیات ممیز شناور = ۴

متوسط CPI برای دیگر دستورات = ۱/ ۳۳

احتمال وقوع جذر ممیز شناور = ۰/۲

CPI برای جذر ممیز شناور = ۲۰

فرض کنید که دو انتخاب در طراحی وجود دارد: اولی کاهش CPI برای جذر ممیز شناور به ۲ و دیگری کاهش CPI برای تمام عملیات ممیز شناور به ۲/۵ وجود داشته باشد. اثر این دو انتخاب را در کارآیی پردازنده مقایسه کنید.

پاسخ: ابتدا فرض می‌کنیم که فقط CPI کاهش می‌یابد و دیگر پارامترهای سیستم بدون تغییر باقی می‌مانند. ابتدا CPI پردازنده اصلی بدون تغییرات را محاسبه می‌کنیم:

$$\text{CPI}_{\text{اصلی}} = \sum_{i=1}^{n} CPI_i \times \left( \frac{IC_i}{\text{تعداد کل دستورالعمل‌ها}} \right)$$

$$= (۴ * ٪۲۵) + (۱/۳۳ * ۰/۷۵) = ۲/۰$$

محاسبهٔ CPI برای حالت بهبود یافته عملیات جذر ممیز شناور با تفریق کردن تعداد سیکل‌های صرفه جویی شده بدست می‌آید:

CPI با جذر بهبود یافته = CPI اصلی) − ۰/۲ * (CPI با جذر قبلی − CPI با جذر جدید )

$$= ۲/۰ - ٪۲ * (۲۰ - ۲) = ۱/۶۴$$

همچنین می‌توان CPI را برای عملیات ممیز شناور با جمع کردن CPI برای عملیات ممیز شناور و عملیات دیگر بدست آورد:

CPI با ممیز شناور جدید = (۰/۷۵ * ۱/۳۳ ) + (۰/۲۵ * ۲/۵) = ۱/۶۲۵

چون CPI برای بهبود تمام واحد ممیز شناور کمی بهتر است پس مناسبتر است که این راه حل انتخاب شود. میزان تسریع برای واحد ممیز شناور جدید بصورت زیر محاسبه می‌شود:

$$\text{میزان تسریع واحد ممیز شناور جدید} = \frac{\text{زمان اجرای قبلی}}{\text{زمان اجرا باواحد ممیز شناور جدید}}$$

$$= \frac{\text{CPI اصلی} \times \text{سیکل کلاک} \times IC}{\text{CPI واحد ممیز شناور جدید} \times \text{سیکل کلاک} \times IC}$$

$$= \frac{\text{CPI اصلی}}{\text{CPI واحد ممیز شناور جدید}} = \frac{2.000}{1.625} = 1.23$$

خوشبختانه، این میزان تسریع برابر با میزان تسریع بدست آمده با قانون امدال در صفحه ۶۱ می‌باشد. معمولاً امکان اندازه‌گیری اجزای مختلف رابطهٔ مربوط به کارآیی پردازنده وجود دارد. این مسأله مزیت اصلی استفاده از رابطهٔ مربوط به زمان پردازنده بر استفاده از قانون امدال است. بخصوص، اندازه‌گیری چیزهایی نظیر درصد زمان اجرا برای دستورالعمل‌های استفاده شده مشکل می‌باشد. در عمل برای محاسبهٔ این رابطه، حاصلضرب تعداد دستورالعمل‌ها در CPI برای هر کدام از دستورالعمل‌ها جمع می‌شود. چون تعداد دستورالعمل و CPI برای هر دستورالعمل جداگانه می‌باشد پس فرمول کارآیی پردازنده مفید می‌باشد.

## اندازه‌گیری و مدل سازی اجزای فرمول کارآیی پردازنده

برای استفاده از فرمول کارآیی پردازنده بعنوان یک ابزار طراحی، باید قادر باشیم که فاکتورهای مختلف آن را اندازه‌گیری کنیم. برای یک پردازنده موجود، اندازه‌گیری زمان اجرا ساده بوده و سرعت کلاک نیز مشخص می‌باشد. چالش اصلی کشف تعداد دستورالعمل‌ها و CPI می‌باشد. پردازنده‌های جدید شمارنده‌هایی برای شمارش تعداد دستورالعمل‌های اجرا شده و تعداد سیکل کلاک دارند. با

بررسی متوالی این شمارنده‌ها می‌توان زمان اجرا و تعداد دستورالعمل‌های اجرا شده در قسمتی از کد برنامه را پیدا کرد، این اطلاعات برای برنامه نویسان مفید بوده و می‌توانند به کمک آنها کارآیی برنامه‌های خود را افزایش دهند. بعضی مواقع، یک طراح یا برنامه نویس مایل است که کارآیی را با دقتی بیشتر از آنچه شمارنده‌های سخت افزاری مشخص می‌کنند، داشته باشد. مثلاً می‌خواهد علت اینکه چرا CPI مقدار فعلی دارد را کشف کند. در چنین مواردی تکنیک‌های شبیه سازی مانند آنچه در طراحی یک پردازنده جدید کاربرد دارد، استفاده می‌شود.

بطور کلی سه کلاس مختلف از تکنیک‌های شبیه‌سازی استفاده می‌شوند. در حالت کلی تکنیک‌های پیچیده‌تر، دقت بیشتری بخصوص برای معماری‌های جدیدتر، دارند و این به بهای زمان اجرای طولانی‌تر می‌باشد. اولین و ساده‌ترین تکنیک و در نتیجه کم هزینه‌ترین مدل، مدل سازی ایستا بر حسب رفتار است. در این تکنیک رفتار پویای برنامه در زمان اجرا، که مشخص کننده میزان اجرای هر دستورالعمل می‌باشد، با یکی از سه روش زیر بدست می‌آید:

۱- با استفاده از شمارنده‌های سخت افزاری موجود در پردازنده که بصورت دوره‌ای ذخیره می‌شوند. این تکنیک معمولاً تخمینی از رفتار بوده ولی دقت آن کم می‌باشد.

۲- با اجرای کدهای‌اندازه‌گیر، در این روش کدهای اندازه‌گیر کامپایل می‌شوند. این کدها اجرا شده و سپس مقدار شمارنده‌ها بررسی می‌شود، مقدار شمارنده‌ها رفتار واقعی را مشخص می‌کنند. (همچنین این روش را می‌توان برای دنبال کردن آدرس‌های مورد استفاده حافظه استفاده کرد که برای تکنیک‌های دیگر شبیه‌سازی مفید می‌باشد.)

۳- با عرضهٔ برنامه در سطح دستورالعمل و شمارش دستورالعمل‌های موجود در فرآیند.

وقتی که نحوه رفتار مشخص شد، می‌توان برنامه را بصورت ایستا با نگاه کردن به کد آن بررسی کرد. مشخص است که در این صورت بدست آوردن تعداد دستورالعمل‌ها ساده است. همچنین می‌توان ترکیب پویای دستورالعمل‌ها را با بیان اینکه میزان تکرار هر نوع دستورالعمل چقدر است، بدست آورد. نهایتاً برای پردازنده‌های ساده می‌توان CPI را بطور تقریبی محاسبه کرد. برای تقریب می‌توان هر بلوک از برنامه یا قطعه‌ای از کدهای برنامه را مدل سازی و بررسی کرده و سپس CPI کلی یا زمان اجرای کلی را با ضرب کردن زمان تخمین زده شده برای هر واحد در میزان تکرار آن واحد بدست آورد. گرچه این نوع مدل سازی رفتار حافظه را نادیده می‌گیرد و محدودیت‌های جدی برای مدل سازی خط لوله‌های پیچیده دارد ولی روش معقول و سریعی برای مدل سازی کارآیی خط لوله‌های اعداد صحیح و کوتاه بدون در نظر گرفتن رفتار حافظه می‌باشد.

روش شبیه سازی نگاه‌داری سابقه (Trace-driven) تکنیک مناسبتری بوده و برای ارزیابی بخصوص مدل سازی کارآیی سیستم حافظه می‌باشد. در این روش سابقه‌ای از نقاط حافظه‌ای که به آنها مراجعه شده ایجاد می‌شود، این کار با شبیه سازی یا با توجه به دستور اجرا شده انجام می‌شود. این سابقه شامل دستور العمل اجرا شده ( که توسط آدرس دستور العمل مشخص می‌شود) و داده هایی که به آنها دسترسی شده ، می‌باشد.

روش شبیه‌سازی نگاه‌داری سابقه به چند روش استفاده می‌شود. کاربرد عمومی‌تر، مدل کردن کارآیی سیستم حافظه می‌باشد که با شبیه سازی سیستم حافظه حاصل می‌شود، این کار شامل شبیه‌سازی حافظه نهان و هر سخت افزار دیگر مدیریت حافظه با دنبال کردن آدرس‌ها می‌باشد. روش شبیه سازی نگاه‌داری سابقه برای سیستم حافظه را می‌توان با آنالیز ایستای کارآیی خط لوله ترکیب کرد تا مدل ساده

و معقولی از فرآیند ساده خط لوله بدست آید. برای خط لوله‌های پیچیده‌تر داده‌های مربوط به سابقه می‌تواند برای آنالیز مشروح کارآرایی خط لوله با شبیه سازی فرآیند خط لوله استفاده می‌شود. چون داده‌های مربوط به سابقه، ترتیب واقعی دستورالعمل ها را نشان می‌دهد، با این روش می‌توان نتایج دقیقتری در مقایسه با روش ایستا بدست آورد. روش شبیه سازی نگاه داری سابقه می‌تواند هر رفتار خط لوله را از رفتار سیستم حافظه مجزا کند. در واقع فرض می‌شود که سوابق کاملاً مسقل از رفتار سیستم حافظه می‌باشد، ولی این مورد برای اکثر پردازنده‌های مدرن صادق نبوده و نیاز به تکنیک سومی می‌باشد.

تکنیک سوم که از همه دقیقتر و پر هزینه‌تر می‌باشد، شبیه سازی متناسب با اجرای برنامه(Execution driven) می‌باشد. در شبیه سازی اجرایی، شبیه سازی مفصل سیستم حافظه و خط لوله پردازنده با یکدیگر انجام می‌شود. این کار مدل واقعی رابطه بین آنها را نشان داده ولی این مسئله بحرانی می‌باشد.

## اصل محلی بودن (Principle of locality)

گرچه قانون امدال قضیه‌ای است که برای هر سیستمی قابل اعمال می‌باشد ولی ملاحظات مهم دیگری هستند که وابستگی به مشخصات برنامه‌ها دارند. مهم‌ترین خصوصیت یک برنامه که معمولاً کشف می‌شود اصل محلی بودن می‌باشد: یعنی برنامه معمولاً داده یا دستورالعملی را که اخیراً استفاده کرده، مجدداً استفاده می‌کند. یک تقریب کلی این است که برنامه ۹۰٪ وقت خود را روی ۱۰ ٪ از کد برنامه صرف می‌کند. مزیت محلی بودن این است که می‌توان با دقت خوبی پیش بینی کرد که چه دستورالعمل ها یا داده‌های حافظه ای در آینده نزدیک با توجه به گذشته نزدیک، استفاده خواهند شد.

اصل محلی بودن دربارهٔ دسترسی به داده‌ها نیز صادق است ولی به قوت دسترسی به کد نیست. دو نوع محلی بودن قابل ملاحظه می‌باشد.محلی بودن زمانی(Temporal locality) یعنی عناصری که اخیراً

استفاده شده‌اند، در آینده نزدیک دوباره استفاده خواهند شد و محلی بودن زمانی (Spatial locality) یعنی وقتی به داده‌ای دسترسی می‌شود به همسایگان نزدیک آن نیز دسترسی خواهد شد.

## مزایای موازی‌گری

بهره بردن از مزایای موازی‌گری یکی از روش‌های مهم برای بهبود کارآیی می‌باشد. در اینجا سه مثال خلاصه بیان می‌شود. اولین مثال استفاده از موازی‌گری در سطح سیستم می‌باشد. برای بالا بردن بازده کارآیی در یک برنامه ارزیابی سرویس دهنده مانند SPEC Web یا TPC می‌توان از یک سیستم با چندین پردازنده و چندین دیسک استفاده کرد. بنابراین بار کاری بین چند پردازنده یا چند دیسک تقسیم شده و بازده کلی افزایش می‌یابد. به همین دلیل مقیاس پذیری برای سرویس دهنده ها معیار با ارزشی می‌باشد.

در سطح هر پردازنده مجزا استفاده از موازی‌گری در سطح دستورالعمل‌ها برای بدست آوردن کارآیی بالاتر حیاتی می‌باشد. یکی از ساده‌ترین کارها برای رسیدن به این مقصود استفاده از خط لوله (pipeline) می‌باشد. ایده اصلی خط لوله هم‌پوشانی اجرای دستورالعمل‌ها می‌باشد، این کار باعث کاهش زمان کلی برای اجرای دنباله‌ای از دستورالعمل‌ها می‌شود. با نگاه کردن از دید رابطهٔ کارآیی پردازنده، می‌توان تصور کرد که خط لوله باعث کاهش CPI از طریق هم پوشانی کلاک‌ها می‌شود. یک نکته کلیدی که باعث می‌شود خط لوله کارا باشد این است که همهٔ دستورالعمل ها به نتایج دستورالعمل قبلی خود وابسته نیستند بنابراین اجرای موازی دو دستور العمل به طور کامل یا جزئی امکان پذیر می‌باشد.

موازی‌گری می‌تواند در سطح مدارات منطقی باشد. مثلاً حافظه‌های نهان شرکت پذیر مجموعه‌ای (Set- associative cache) از چندین بانک حافظه استفاده می‌کنند و معمولاً جستجوی یک داده همزمان

در آنها انجام می‌شود . واحدهای ALU پیشرفته از پیش بینی بیت نقلی (Carry – lookahead)استفاده می‌کنند که باعث می‌شود محاسبات به جای وابستگی خطی به طول بیت‌ها، وابستگی لگاریتمی به طول بیت ها داشته باشد .

طراحان به روشهای مختلفی از مزایای موازی‌گری سود می‌برند. یکی از تکنیک‌های مرسوم اجرای موازی ۲ یا ۳ درخواست ورودی و سپس انتخاب با تأخیر آنها می‌باشد. این تکنیک در جمع‌کننده‌ها، حافظه‌های شرکت پذیر اشتراکی و حل مسئله پرش در خط لوله استفاده می‌شود.