به نام خدا

تهیه کننده: ابراهیم صدیقی شماره دانشجویی: ۹۹۳۱۰۹۸

سوال۱)
الف) یک CPU 10 گیگاهرتز با چندین چالش روبرو خواهد شد که مانع تحقق آن تا سال 2005 شد. یکی از موانع اصلی مسئله مصرف برق و اتلاف گرما است. با افزایش سرعت ساعت، مصرف برق نیز افزایش می‌یابد که منجر به تولید گرمای بیشتر می‌شود. در فرکانس 10 گیگاهرتز، انرژی مورد نیاز و اتلاف گرما بسیار زیاد است و به راه حل های خنک کننده پیشرفته مانند خنک کننده مایع یا مواد عجیب و غریب نیاز دارد که در آن زمان امکان پذیر یا مقرون به صرفه نبودند. علاوه بر این، مقیاس بندی Dennard، اصلی که نشان می دهد با کوچک شدن ترانزیستورها، چگالی توان آنها ثابت می ماند، نقش مهمی ایفا کرد. با این حال، با رسیدن ترانزیستورها به اندازه های نانومقیاس، پوسته پوسته شدن دنارد شروع به شکسته شدن کرد. این خرابی منجر به افزایش نشت نیرو و اتلاف گرما شد و حفظ سرعت کلاک پیش‌بینی‌شده 10 گیگاهرتز بدون به خطر انداختن راندمان و قابلیت اطمینان را به چالش کشید. انتقال از تراشه های اولیه ~ 500 کیلوهرتز به تراشه های فعلی ~ 5 گیگاهرتز به دلیل چندین عامل کمتر مشکل ساز بود. اولاً، پیشرفت‌ها در فناوری نیمه‌رساناها امکان مدیریت بهتر توان و مکانیسم‌های اتلاف گرما را فراهم کرد. به‌علاوه، تغییر معماری‌های چند هسته‌ای، CPUها را قادر می‌سازد تا بدون تکیه بر افزایش سرعت ساعت، به عملکرد بالاتری دست یابند. با موازی سازی وظایف در چندین هسته، طراحان CPU می توانند عملکرد را بدون وابستگی صرفاً به بهبود سرعت ساعت افزایش دهند.

ب)علارغم محدودیت‌هایی که در افزایش قابل توجه سرعت کلاک پردازنده وجود دارد، طراحان استراتژی‌های مختلفی را برای افزایش عملکرد اتخاذ کرده‌اند. یک رویکرد کلیدی از طریق بهینه سازی معماری است. با اصلاح ریزمعماری پردازنده‌ها، طراحان می‌توانند خطوط لوله دستورالعمل، سلسله‌مراتب حافظه پنهان و

مکانیسم‌های پیش‌بینی شاخه را برای افزایش کارایی بدون تکیه بر افزایش سرعت ساعت بهبود دهند. علاوه بر این، پیشرفت‌ها در محاسبات موازی در افزایش عملکرد CPU مؤثر بوده است. پردازنده های چند هسته ای امکان اجرای همزمان وظایف را فراهم می کنند و از موازی سازی برای دستیابی به توان عملیاتی و کارایی بالاتر استفاده می کنند. با تقسیم بار کاری بین چندین هسته، طراحان CPU می توانند از موازی سازی برای بهبود عملکرد کلی سیستم استفاده کنند.

سوال۲)
الف)
SISD: در سیستم‌های SISD، یک پردازنده واحد یک جریان دستورالعمل را اجرا می‌کند تا در یک زمان روی یک جریان داده کار کند. سیستم های تک پردازنده سنتی در این دسته قرار می گیرند.

SIMD: سیستم های SIMD از چندین عنصر پردازشی برای انجام یک عملیات مشابه بر روی چندین نقطه داده به طور همزمان استفاده می کنند. این معماری در پردازنده‌های گرافیکی رایج است، جایی که یک دستورالعمل واحد به واحدهای پردازشی متعدد برای پردازش عناصر داده‌های مختلف به طور همزمان پخش می‌شود.

MISD: سیستم‌های MISD شامل چندین پردازنده هستند که دستورالعمل‌های متفاوتی را بر روی یک جریان داده اجرا می‌کنند. این معماری به دلیل موارد استفاده محدود در کاربردهای عملی نادر است.
MIMD: سیستم‌های MIMD دارای چندین پردازنده هستند که به طور مستقل عمل می‌کنند و دستورالعمل‌های مختلفی را روی جریان‌های داده‌های مختلف به طور همزمان اجرا می‌کنند. این معماری در CPU های چند هسته ای و سیستم های محاسباتی توزیع شده رایج است.

ب)
GPU: پردازنده‌های گرافیکی عمدتاً از معماری SIMD استفاده می‌کنند. آنها داده های گرافیکی را با اجرای دستورالعمل های رندر گرافیکی مشابه در چندین نقطه داده به طور همزمان پردازش می کنند. این موازی سازی عملکرد گرافیکی و توان محاسباتی را افزایش می دهد.

CPU چند هسته ای: پیکربندی CPU شرح داده شده با معماری MIMD هماهنگ است. هر هسته در CPU چند هسته ای می تواند به طور مستقل دستورالعمل های مختلفی را بر روی جریان های داده مختلف اجرا کند. طراحی خط لوله، توان عملیاتی دستورالعمل را افزایش می دهد، در حالی که پسوند AVX2 از عملیات SIMD برای پردازش برداری بهینه پشتیبانی می کند.

ریزپردازنده 8 بیتی: ریزپردازنده داده شده تحت معماری SISD قرار می گیرد. با پردازش تک هسته ای یک دستور در یک زمان در یک جریان داده، به صورت متوالی بر روی داده های 8 بیتی عمل می کند. این معماری ساده برای کارهای کم پیچیدگی که نیازی به پردازش موازی نیست مناسب است.

سوال ۵)
سیستم‌های ccNUMA (دسترسی به حافظه غیریکنواخت منسجم کش) یک نوع تخصصی از معماری NUMA هستند که انسجام کش را در چندین هسته در یک سیستم چند پردازنده تضمین می‌کند. در یک سیستم ccNUMA، هر پردازنده دارای حافظه محلی و کش مخصوص به خود است، اما می تواند به حافظه و حافظه نهان دیگر پردازنده ها نیز دسترسی داشته باشد. اصل کلیدی ccNUMA حفظ انسجام کش است و اطمینان حاصل می کند که همه هسته ها دید ثابتی از حافظه دارند. در سناریوی داده شده، جایی که یک متغیر مشترک بین دو هسته به طور مداوم تغییر می کند و در حافظه نهان سطح پایین منحصر به فرد هسته هر دو هسته ذخیره می شود، عوامل متعددی وارد بازی می شوند که می توانند عملکرد برنامه را به طور قابل توجهی تحت تاثیر قرار دهند. هنگامی که هر دو هسته متغیر مشترک را اصلاح می کنند، مکانیسم های انسجام کش باید

اطمینان حاصل کنند که تغییرات ایجاد شده توسط یک هسته برای هسته دیگر قابل مشاهده است. این شامل پروتکل‌های ابطال کش و انسجام حافظه پنهان برای همگام‌سازی داده‌ها در حافظه پنهان است. تغییر مداوم متغیر مشترک توسط چندین هسته می تواند منجر به کشمکش کش و ترافیک انسجام شود. کشمکش کش زمانی رخ می دهد که چندین هسته سعی می کنند به طور همزمان به یک خط کش یکسان دسترسی یا اصلاح کنند که منجر به تاخیر و کاهش عملکرد می شود. ترافیک انسجام به سربار ارتباطی اطلاق می شود که برای حفظ انسجام حافظه پنهان انجام می شود، که می تواند با افزایش فرکانس تغییرات در متغیر مشترک افزایش یابد. برای کاهش تأثیر عملکرد تغییرات متغیر مشترک در یک سیستم ccNUMA، توسعه‌دهندگان می‌توانند از تکنیک‌هایی مانند برنامه‌نویسی آگاه از حافظه پنهان، بهینه‌سازی محل داده‌ها و به حداقل رساندن دسترسی به داده‌های مشترک غیرضروری استفاده کنند. با کاهش کشمکش کش، بهینه سازی الگوهای دسترسی به داده ها و به حداقل رساندن ترافیک انسجام، عملکرد برنامه های در حال اجرا بر روی سیستم های ccNUMA را می توان بهبود بخشید.

سوال ۶)
قانون امدال را می توان به صورت زیر تدوین کرد:

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

- speedup بهبود نظری در عملکرد است.
- P نسبتی از برنامه است که می تواند موازی شود.
- N تعداد پردازنده ها است.

قانون گوستافسون افزایش سرعت را تخمین می زند که به صورت زیر بدست می آید:

$$\begin{aligned} S &= s + p \times N \\ &= s + (1 - s) \times N \\ &= N + (1 - N) \times s \end{aligned}$$

- S سرعت تئوری برنامه با موازی سازی است.
- N تعداد پردازنده‌ها است.
- s و p کسری از زمان صرف شده برای اجرای قسمت های سریال و قسمت های موازی برنامه در سیستم موازی است، که در آن $s + p = 1$

محدودیتی که قانون گوستافسون به آن پرداخته است:
قانون گوستافسون به محدودیت قانون آمدال می پردازد که اندازه مشکل ثابتی را در نظر می گیرد. قانون آمدال بر حجم کار ثابت و تأثیر موازی سازی بر آن حجم کار تمرکز دارد. در مقابل، قانون گوستافسون بر افزایش اندازه مشکل با افزایش تعداد پردازنده‌ها تأکید می‌کند و امکان رویکرد مقیاس‌پذیرتر را فراهم می‌کند.

مقایسه مقیاس پذیری:
با فرض ثابت ماندن هزینه قسمت سریال الگوریتم ها، بیایید مقیاس پذیری یک الگوریتم O(n) را با الگوریتم O(3^n) مقایسه کنیم:

برای الگوریتم O(n)، مقیاس پذیری خطی است، به این معنی که با بزرگ شدن اندازه مسئله، زمان اجرا به طور متناسب افزایش می یابد. افزودن پردازنده های بیشتر منجر به افزایش سرعت خطی می شود.
برای الگوریتم O(3^n)، مقیاس پذیری مکعب است، که نشان می دهد با افزایش اندازه مسئله، زمان اجرا به طور قابل توجهی سریعتر رشد می کند. در حالی که افزودن پردازنده‌های بیشتر همچنان می‌تواند عملکرد را بهبود بخشد، بازده کاهش یافته به دلیل پیچیدگی مکعبی، سرعت را در مقایسه با مقیاس‌پذیری خطی الگوریتم O(n) محدود می‌کند.