تمرین اول os

«به نام خدا»

امیرمحمد پیرحسینلو

9531014

① DMA از جهت دارد، زیرا:

اگر cpu به ماشینه کردن کار بپردازد (انتقال داده)، آنگاه کار مفیدی انجام نمی‌دهد و از کارایی سیستم کاسته می‌شود. همچنین ممکن است cpu به داده‌ای نیاز داشته باشد که هم اکنون در دیسک است و باید به RAM تحویل داده شود. البته همچنان، ساختار cpu موجود در سیستم بستگی دارد، برای مثال در سیستم‌های توزیع شده (embeded) به احتمال خیلی زیاد دسترسی cpu به حافظه نسبت به DMA، اولویت دارد زیرا اگر cpu مدت زمان طولانی بیکار بماند، عملیات کار بر انجام نشده و نارضایتی پیش می‌آید.

②

A) programmed I/O: در این روش، برنامه‌ای که در cpu در حال اجراشدن است، از instruction ها برای انتقال داده استفاده می‌کند. این روش در تضاد با روش DMA است، زیرا در روش DMA، cpu نقشی در انتقال داده ندارد. از این روش برای انتقال داده بین network adapter و storage device ATA استفاده می‌شود.

B) Interrupt driven I/O: در این روش، دستگاه‌های IO ~~به طور مداوم~~ برای ارسال داده به cpu، سیگنالی را تحت عنوان interrupt به cpu ارسال می‌کنند، cpu به طور مداوم در حال اجرای instruction ها است، به محض رسیدن interrupt، cpu دستور العمل فعلی را انجام داده

پس سراغ داده‌ای که توسط دستگاه IO به cpu فرستاده شده می‌رود. cpu به وسیله سگنال interrupt از آمدن دیتا توسط IO مطلع شده است. کارآیی بیشتری نسبت به روش polling دارد و performance سی‌پی‌یو را نسبت به polling افزایش می‌دهد.

ج) DMA : در این روش انتقال داده، وظیفه انتقال داده از cpu به یک device manager سپرده می‌شود. این دستگاه انتقال داده از هبدا به مقصد را انجام داده سپس کنترل را به cpu برمی‌گرداند. مثلاً cpu به آن می‌گوید x بایت دیتا از خانه‌ی y در حافظه را به خانه‌ی z در آن پیرو دستگاه این کار را انجام بشروع می‌دهد و دوباره cpu مشغول کار می‌شود. معمولا از این روش در انتقال داده با حجم زیاد استفاده می‌شود.

③ می‌توان طول برنامه‌هایی که با رسیدن interrupt اجرا می‌شوند را کوتاه‌تر کرد تا سریع‌تر اجرا شوند. در این صورت، پیچیدگی وقفه‌های تودرتو کمتر می‌شوند.

می‌توان interrupt limit تعریف کرد و از ایجاد پیچیدگی‌های زیاد در وقفه‌های تودرتو جلوگیری کرد. اما باید یک queue در نظر گرفت را interrupt های اضافی به داخل آن بروند و بعدا به آن‌ها رسیدگی بشود.

④ الف) SISD : یک cpu، تنها یک جریان دستورالعمل دارد که آن‌ها را با توجه به داده‌ی موجود در تنها Main Memory خود اجرا می‌کند. این معماری همان معماری نومن نیومن است. single instruction, single data

ب) SIMD : شامل یک جریان دستورالعمل ورودی و چند جریان ورودی داده از Main Memory است. در این دستگاه، پردازنده یک دستورالعمل را روی چندین ورودی داده اعمال می‌کند (یک دستورالعمل یکسان). در این کامپیوترها، data level parallelism داریم نه concurrency. در هر لحظه تنها یک process در حال اجرا است. (البته single core / cpu در نظر می‌گیریم)
single instruction, multiple data.

ج) MISD : چندین جریان دستورالعمل ورودی و تنها یک جریان داده‌ی ورودی وجود دارد، پس عملیات های متفاوت بر روی یک داده انجام می‌شود، به عبارتی روی داده‌ی یکسان. concurrency در این کامپیوترها وجود دارد و معماری pipeline هم بر این تایپ متعلق است. multiple instruction, single data

د) MIMD : چندین جریان دستورالعمل و داده وجود دارد. هسته‌های پردازنده‌ها به طور مستقل و ناهمگام با یکدیگر روی داده‌های متفاوت عملیات انجام می‌دهند. می‌توان در این کامپیوترها parallelism و concurrency را مشاهده کرد.
multiple instruction, multiple data

⑤ مود هسته : حالتی است که سیستم عامل کنترل جریان اجرا را برعهده دارد (به عبارتی برنامه‌های داخل سیستم عامل درحال اجرا هستند و برنامه‌ای از جانب کاربر اجرا نمی‌شود.)

مود کاربر: حالتی که برنامه‌ی کاربر توسط cpu اجرا می‌شود.

تفاوت این دو حالت از طریق flag مشخص می‌شود.

درحالت (مود) کاربر، کاربر نباید بتواند کد موجود برای سیستم عامل را تغییر دهد. کاربر نباید بتواند فضای حافظه که به cpu و سایر برنامه‌ها اختصاص یافته است را تغییر دهد و سیستم عامل های امروزی با نظارت جلوی این کار را می‌گیرند.

⑥ وقفه سخت افزاری: وقفه‌های IO که از دستگاه‌های IO مثل mouse و keyboard و... به cpu فرستاده می شوند. یا مثلاً # وقفه‌ای که timer برای حلقه های بی نهایت می فرستد.

وقفه نرم افزاری: divide by zero accessing unavailable memory ...