بسم الله الرحمن الرحیم

پروژه نهایی درس معماری کامپیوتر

استاد بیت الهی

اعضای گروه:

محمد حسین میرزایی

متین محمود خانی

محمد حسین حسنی

دانشکده کامپیوتر دانشگاه علم و صنعت

Processor

کاری که پروسسور انجام می دهد، این است که به ما یک virtual address بدهد. پس کدی که باید پیاده سازی شود باید به ما یک آدرس مجازی دهد. این آدرس ها در یک فایل ذخیره شده است (برای ساده تر شدن کار، آدرس ها در آرایه ذخیره شده اند). با استفاده از index آن، یکی از آدرس های مجازی را انتخاب می کنیم.

storage

در storage دیتا های ما قرار می گیرند و این دیتا ها به صورت دسته دسته (page) در کنار یکدیگر قرار گرفته اند. چون داده های ما در نهایت در cache به صورت two-word هستند، پس داده ها در ابتدا نیز در page ها به صورت two-word ذخیره شده اند (برای راحتی کار، two-word را 64 بیتی در نظر گرفته و در انتها موقع استفاده از آن، با کمک گرفتن از index ها آن را به دو word جدا تبدیل کرد.) چون page size ما 128 بایتی (word 32) می باشد و همچنین دیتا ها باید به صورت twp-word ذخیره شوند، درنتیجه آن را به صورت 16 ردیف شامل 2 ورد قرار می دهیم. همچنین ظرفیت مموری ما 512 ورد می باشد و از آنجا که page های ما 32 وردی هستند، پس حداقل نیاز هست که حداقل 8 عدد از این page ها داخل دیسک وجود داشته باشد. و در صورت miss شدن page (page fault) در رم، یکی از page های موجود در دیسک، جایگزین آن در مموری خواهد شد.

Memory

مموری از دو بخش تشکیل شده است. یک بخش مربوط به دیتا های ذخیره شده و بخش دیگر مربوط به page table می باشد. برای page table یک انتیتی جداگانه در نظر گرفته شده است. با توجه به این که page size ما 128 بایت است، 7 بیت برای نمایش آن نیاز است (2^7 = 127).حال با توجه به این که virtual address ما 16 بیتی است و طبق مطالب گفته شده، 7 بیت برای page offset نیاز است، پس 9 بیت برای vpn باقی می ماند. با توجه به فرض مسیله، همه vpn ها (با توجه به مقدار آدرس) به یکی از ردیف های page table اشاره می کنند. پس با توجه به آن که 9 بیت برای نمایش vpn نیاز است، پس در کل page table از 2^9 ردیف تشکیل شده است.حال برای محاسبه مقدار کل اندازه page table، باید تعداد بیت های یک ردیف آن در کل ردیف های آن ضرب شوند. همچنین یک ردیف page table از valid bit و ppn تشکیل شده است.valid bit تک بیتی و ppn را 7 بیتی در نظر می گیریم (مقدار ppn باید متناسب با بلاک های دیتا در رم باشد.مقدار دقیق تر برای تعداد بیت ppn برابر 4 می باشد.اما برای آن که محاسبات ما رند تر شود و اعداد ما به صورت مضارب 2 ذخیره شود،آن را 7 بیتی در نظر گرفته که مجموعا با بیت valid 8 بیتی شود.) پس دیتای موجود در هر ردیف 8 بیتی می باشد. پس 2^9 را در 2^3 ضرب می کنیم تا حجم page table به دست آید که برابر 2^12 بیت می شود.2^12 بیت برابر با 2^7 ورد می باشد که 128 ورد می شود.کل مموری ما شامل 512 ورد می

تواند باشد.حال اگر 128 را از 512 کم کنیم، به 384 ورد می رسیم که حافظه ای است که به دیتا تعلق دارد. Page هایی که از دیسک به مموری می آیند، شامل 32 ورد می باشند.پس در 384 ورد باقی مانده در مموری، می توان 12 page ذخیره کرد. مموری دو کار را انجام می دهد.یکی خواندن دیتا از دیسک و دیگری نوشتن دیتا در cache.پس ما برای هر خانه از مموری،یک check bit در نظر می گیریم تا بفهمیم کاری که مموری باید با آن خانه انجام دهد، کدام است.

Page Table

همانطور که در بخش های قبلی گفته شد،page table مانند یه دیکشنری است که برای هر vpn یک ppn نسبت می دهد.با توجه به فرض ما، همه ی vpn ها در page table موجود است و همه ی مقادیر آن hit می شود.با توجه به این که دیتا های page table با دیتا های ساخته شده توسط توسط ppn برابر است، به تعداد لازم ppn تولید کرده و در هر یک از خانه های page table قرار می دهیم.در این مرحله ورودی ما یک virtual address می باشد که از دو بخش vpn و page offset ساخته شده است و با index بندی مناسب page offset و vpn را جدا کرده و با دادن vpn به page table، ppn نظیر آن را پیدا می کنیم.سپس با در کنار هم قرار دادن ppn و page offset ،آدرس فیزیکی را تولید می کنیم.

Cache

در این قسمت، مانند قسمت مموری،چک بیتی وجود دارد که بفهمیم دیتا در cache وجود دارد و آن را به خروجی دهیم.در غیر این صورت باید دیتا از مموری

خوانده شود و در cache نوشته شود.حال برای آن که دیتا به خروجی داده شود،چند مرحله نیاز است.طبق کاربرد cache ابتدا یک آدرس فیزیکی به بخش های مختلف نظیر byte offset، word offset،index و tag شکسته شود.چون cache ما 32 ردیف دارد، پس با 5 بیت که قسمت index را تشکیل می دهد، می توان آن را ساخت.سپس تعداد بیت های physical address را به دست آورده و هر یک از قسمت های گفته شده را بررسی می کنیم. همچنین physical address ما از page offset و ppn تشکیل شده است.هر کدام از این دو 7 بیتی هستند.در نتیجه physical address 14 بیتی خواهد بود.byte offset ما دوبیتی و word offset ما یک بیتی می باشد.همچنین index 5 بیتی می باشد.در نتیجه 6 بیت باقی مانده برای tag می باشد.حالا برای پیدا کردن دیتا،index مورد نظر در خانه cache را بررسی می کنیم.اگر valid bit آن صفر باشد،مقدار آن را یک کرده و به مموری مراجعه کرده و دیتای مورد نظر را بازخوانی می کنیم و مقدار بیت hit را صفر در نظر می گیریم.اما اگر صفر نبود،قسمت tag از physical address را با قسمت tag در خانه های cache مقایسه می کنیم و اگر مقدار آن دو برابر بود، از میان دو ورد موجود در ردیف cache ،با توجه به word offset یکی را انتخاب کرده و با توجه به byte offset بایت مورد نیاز از ورد مورد نظر را استخراج می کنیم و در خروجی نمایش می دهیم.این روش ،پیاده سازی cache به صورت direct map بود.در پیاده سازی به روش set 2،در حالت کلی، تعداد ردیف های cache نصف شده و به جای 32 خانه،دیتا ها در 16 خانه ذخیره می شوند.البته اندازه هر بخش دو برابر شده و به جای ذخیره 2 ورد و یک

tag در index مورد نظر قابلیت ذخیره 2 دسته از دیتا های 2 وردی را خواهد داشت.