BIMS : ساختار توکار میانی حافظه برای بهبود حافظه‌های تغییر فاز چندسطحی

سید صابر نبوی لاریمی[۱]، کارشناسی ارشد؛ مهدی کمال[۲]، استادیار؛ علی افضلی کوشا[۳]، استاد

۱- دانشکده مهندسی برق و کامپیوتر - دانشکده فنی - دانشگاه تهران - تهران - ایران - s_nabavi@ut.ac.ir
۲- دانشکده مهندسی برق و کامپیوتر - دانشکده فنی - دانشگاه تهران - تهران - ایران - mehdikamal @ut.ac.ir
۳- دانشکده مهندسی برق و کامپیوتر - دانشکده فنی - دانشگاه تهران - تهران - ایران - afzali@ut.ac.ir

چکیده: در این مقاله، روشی به نام ساختار توکار میانی حافظه (BIMS) را معرفی خواهیم کرد که باعث کاهش انرژی مصرفی و زمان دسترسی حافظه‌های اصلی ساخته‌شده با فناوری حافظه‌های تغییر فاز (PCM) خواهد شد. این روش از قابلیت افزاره‌های PCM که قادر هستند هم به‌صورت سلول تک‌سطحی (SLC) و هم چندسطحی (MLC) مورد استفاده قرار بگیرند، استفاده می‌کند. در این روش، داده‌ها به‌صورت پیش‌فرض در سلول‌هایی با قابلیت ذخیره‌سازی بیشتر از یک بیت ذخیره می‌شوند. اما مکانیزم داخلی این روش، سلول‌های صفحات فیزیکی بلا استفاده را به سلول‌های تک‌سطحی تبدیل می‌کند. با استفاده از این صفحات، لایه‌ای بین حافظه‌ی نهان پردازنده و صفحات اصلی به‌وجود می‌آورد که می‌تواند دستورات خواندن و نوشتن در حافظه را در مدت زمان کمتر و با انرژی کمتر پاسخ دهد. این لایه میانی، بسیاری از دسترسی‌ها به صفحات با افزاره‌های MLC را با جذب آن‌ها از بین می‌برد.

واژه‌های کلیدی: حافظه تغییر فاز (PCM)، مدیریت حافظه، صفحه، سلول حافظه چندسطحی، زمان دسترسی، انرژی مصرفی.

BIMS: Built-in Intermediate Memory Structure to Improve Multi-Level Phase Change Memories

S. S. Nabavi Larimi[1], MSc; M. Kamal[2], Assistant Professor; A. Afzali-Kusha[3], Professor

1- School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran, Email: s_nabavi@ut.ac.ir
2- School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran, Email: mehdikamal @ut.ac.ir
3- School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran, Email: afzali@ut.ac.ir

Abstract: In this paper, we propose a Built-in Intermediate Memory Structure (BIMS) to improve the efficiency of main memory architectures based on Phase Change Memory (PCM). It exploits the capability of the PCM device which can be used as both multi-level cell (MLC) and single-level cell (SLC) during the processor operation. The proposed structure invokes physical pages with MLC devices for a normal data storage. By utilizing an internal page management mechanism, however, it turns memory cells of some unused physical pages into the SLC mode. BIMS exploits such pages to provide an intermediate layer for the memory read and write requests with better access time and lower energy consumption. This intermediate layer diminishes most of the accesses to the pages with the MLC devices by absorbing most of the incoming memory requests.

Keywords: Phase change memory; memory managment; page; multi-level cell, access time, energy consumption.

تاریخ ارسال مقاله: ۱۳۹۵/۱۲/۰۲
تاریخ اصلاح مقاله: ۱۳۹۶/۰۶/۱۱
تاریخ پذیرش مقاله: ۱۳۹۶/۰۸/۱۱
نام نویسنده مسئول: مهدی کمال
نشانی نویسنده مسئول: دانشکده مهندسی برق و کامپیوتر - دانشکده فنی - دانشگاه تهران - تهران - ایران

۱ -مقدمه

نوع حافظه‌های مورد استفاده و مدیریت آن‌ها نقش مهمی در کارایی هر سیستم پردازشی دارد [۱، ۲]. حافظه‌ی اصلی دستگاه‌های امروزی که از فناوری DRAM[۱] استفاده می‌کنند، بخش عظیمی از بودجه‌ی توان مصرفی را به خود اختصاص می‌دهند و در حالی که کوچک‌کردن سایز افزاره‌ها در آن‌ها پیچیده می‌باشد[۳]. در نتیجه، محققین در تلاش‌اند تا جایگزینی برای این فناوری بیابند. یکی از انتخاب‌های اصلی این جایگزینی، فناوری حافظه‌های تغییر فاز (PCM[۲]) است. این فناوری نخستین بار در دهه‌ی ۱۹۶۰ میلادی به‌منظور استفاده در دیسک‌های نوری معرفی شد [۴]. امروزه با به‌کارگیری مواد بهتر و روش‌های پیشرفته‌تر مقیاس‌پذیری، حافظه‌های PCM به یکی از مهم‌ترین گزینه‌های ساخت حافظه‌های اصلی تبدیل شده‌اند[۵].

فناوری PCM از گروهی از مواد با نام عمومی chalcogenide glass برای ذخیره‌سازی داده استفاده می‌کند. هرکدام از این مواد دست‌کم دارای یکی از عناصر گروه شانزدهم جدول تناوبی به غیر از اکسیژن است (شامل گوگرد (S)، سلنیم (Se) و تلوریم (Te)). این مواد دارای دو حالت پایدار اتمی هستند: حالت بلورین[۳] و حالت نامنظم[۴]. در حالت بلورین، تمامی ذرات این مواد در آرایشی منظم کنار هم قرار دارند در حالی که در حالت نامنظم، چنین هماهنگی‌ای برقرار نیست. این گوناگونی ساختاری باعث تفاوت در خواص الکتریکی این مواد نیز می‌شود به‌طوری که در حالت بلورین جریان الکتریکی را همانند یک رسانا منتقل می‌کنند درحالی‌که در حالت نامنظم مقاومتی با مرتبه‌ی بزرگی چند برابر بیشتر از خود نشان می‌دهند [۵].

مواد chalgonide را می‌توان با استفاده از گرما به هرکدام از دو حالت گفته‌شده برنامه‌ریزی کرد. در دیسک‌های نوری این گرما را لیزرها تأمین می‌کنند و در حافظه‌های PCM این عمل با تزریق جریان الکتریکی به یک گرم‌کننده[۵] انجام می‌شود. همین برنامه‌پذیربودن حافظه‌های PCM امکان استفاده از آن‌ها در دستگاه‌های دیجیتال امروزی را فراهم کرده است. این حافظه‌ها نیازی به فرایند بازنویسی[۶] که در DRAM برای جلوگیری از نابودی داده‌ها استفاده می‌شود، ندارند و حتی داده را بدون وجود منبع تغذیه نگاه می‌دارند. همچنین، قدرت مقیاس‌پذیری بسیار خوبی هم در آزمایش‌ها [۶] و هم در صنعت [۷] از خود نشان داده‌اند.

برتری دیگر PCM این است که سلول‌های این فناوری را می‌توان به حالت‌هایی مابین حالات بلورین و نامنظم برنامه ریزی کرد. به این ترتیب، میزان مقاومت هر سلول پس از برنامه‌ریزی در طیفی قرار می‌گیرد که یک طرف آن مقاومت حداکثری حالت نامنظم است و طرف دیگر طیف نیز مقاومت حداقلی حالت بلورین است. چنین حالاتی را می‌توان با کنترل دقیق مقدار، مدت‌زمان و حتی شیب پالس جریان الکتریکی اعمال شده به سلول به‌دست آورد. این حالات اضافه به هر سلول این توانایی را می‌دهند که بیشتر از یک بیت در خود ذخیره کند. هر سلولی که از این قابلیت استفاده کند را MLC[۷] و سلول‌هایی که از این قابلیت استفاده نکنند را نیز SLC[۸] می‌نامند. به این ترتیب MLC ها به هر تراشه قابلیت ذخیره سازی داده با تراکم بیشتر را می‌دهند[۸].

در مورد SLC تنها می‌توان با اعمال یک پالس جریان، هر سلول را به یکی از حالات بلورین یا نامنظم برنامه‌ریزی کرد. اما در مورد MLC رسیدن به حالات میانی تنها با یک پالس ممکن نیست. در عمل برای آنکه بتوان حالات میانی مشابه در همه‌ی سلول‌ها ایجاد کرد، از روش‌هایی بانام عمومی P&V[۹] استفاده می‌شود[۹]. نتیجه اینکه برای نوشتن مقداری داده در سلول‌های MLC تعداد زیادی عملیات نوشتن و خواندن نیاز است. به همین دلیل است که سلول‌های MLC از نظر کارایی و انرژی مصرفی بهینه نیستند. همچنین از آنجایی که عمر سلول‌های PCM به عکس تعداد عملیات نوشتن در آن‌ها بستگی دارد، حافظه‌های MLC عمر کمتری نسبت به SLC خواهند داشت[۱۰].

لازم به ذکر است که ساختار سلول‌های SLC و MLC کاملاً مشابه است و آنچه تفاوت این سلول‌ها را مشخص می‌کند مدارهای جانبی آن‌هاست. به این ترتیب، می‌توان با استفاده از مدارهای جانبی متفاوت، از سلول‌های موجود در یک تراشه‌ی PCM هم به‌صورت SLC و هم به‌صورت MLC استفاده کرد. نتیجه اینکه در مورد تعداد بیت‌هایی که در یک سلول PCM ذخیره می‌شود، حق انتخاب وجود دارد. سیستم (کاربر) می‌تواند تعیین کند که سلول‌ها به شکل SLC و با کارایی بالا استفاده شوند یا به‌صورت MLC که داده‌ی بیشتری ذخیره کنند. این حق انتخاب فرصتی به‌وجود می‌آورد تا بتوان مشکلات هرکدام از این دو حالت را با استفاده از برتری‌های دیگری پوشش داد. در این مقاله روش BIMS[۱۰] را به همین منظور معرفی خواهیم کرد. این روش، بخش‌هایی از حافظه، که به‌طور پیش فرض MLC هستند، را به‌صورت SLC مورد استفاده قرار می‌دهد. این بخش به‌صورت لایه‌ای میان حافظه‌ی اصلی و پایین‌ترین سطح حافظه‌ی نهان عمل می‌کند. در این روش، بخش عظیمی از درخواست‌هایی که برای حافظه فرستاده می‌شوند با دسترسی به بخش SLC پاسخ داده می‌شوند و تنها بخش اندکی نیاز به دسترسی به بخش MLC دارند. در نتیجه، سرعت، توان مصرفی و عمر این حافظه‌ها بهبود می‌یابد. بخش SLC درواقع، آن قسمت از حافظه است که مورد استفاده واقع نشده است و توسط BIMS به کار گرفته‌شده است. صفحاتی از حافظه که به‌تازگی مورد دسترسی قرار گرفته‌اند از بخش MLC به بخش SLC منتقل می‌شوند تا دسترسی‌های احتمالی بعدی به آن‌ها به‌طور بهتری پاسخ داده شوند. به‌خاطر وجود محلی‌بودن[۱۱] دسترسی‌های حافظه، اکثر دستورات را می‌توان تنها با دسترسی به بخش SLC پاسخ داد.

به‌منظور استفاده‌ی بهینه از سلول‌های موجود، BIMS بخش‌های SLC و MLC را به‌صورت پویا تقسیم می‌کند. حجمی از داده که باید به‌صورت MLC ذخیره شود را نیاز سیستم تعیین می‌کند و حجم بخش SLC را نیز BIMS بر اساس تنوع آدرس‌هایی که مورد دسترسی قرار می‌گیرند، مشخص می‌کند. با افزایش نیاز سیستم به حافظه، BIMS بخشی از SLC را به MLC برگردانده و نیاز سیستم را برطرف می‌کند. کارایی BIMS با شبیه‌سازی اندازه‌گیری شده است. نتایج نشان می‌دهند که حافظه‌هایی که از BIMS استفاده می‌کنند، به نسبت

حافظه‌های معمولی PCM، به‌طور متوسط ۸۰٪ انرژی کمتر مصرف می‌کنند و ۲۶٪ سریع‌تر به دسترسی‌ها پاسخ می‌دهند.

در ادامه‌ی این مقاله، در بخش ۲، انگیزه‌ی تحقیق مورد بررسی قرار خواهد گرفت. سپس ساختار BIMS را در بخش ۳ توضیح داده خواهد شد. در بخش ۴، نحوه شبیه‌سازی و نتایج شبیه‌سازی ارائه می‌شود. در نهایت در بخش ۵، نتیجه این مقاله ارائه خواهد شد.

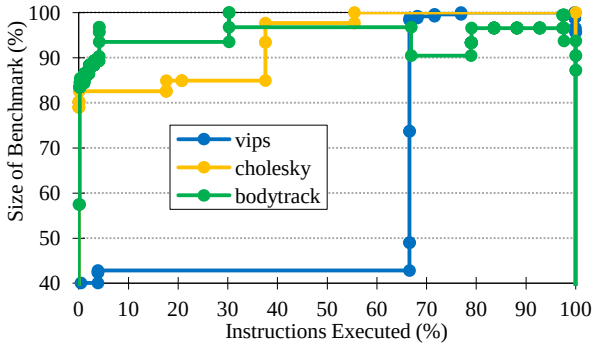
۲ -انگیزه‌ی تحقیق

در این بخش دو موضوع را بررسی می‌کنیم که می‌توانند در بهینه‌کردن حافظه‌هایی که با PCM ساخته می‌شوند به ما کمک کنند. نخست اینکه همه‌ی برنامه‌ها به یک اندازه به حافظه نیاز ندارند. دوم اینکه اکثر برنامه‌ها به حداکثر حافظه‌ی مورد نیاز خود در تمام زمان اجرا نیاز ندارند. اکثر برنامه‌ها در حین اجرا، حافظه را اشغال می‌کنند و سپس آزاد می‌کنند. عموماً سیستم‌ها را برای بدترین حالت ممکن از نظر حجم حافظه‌ی مورد نیاز طراحی می‌کنند. به این ترتیب خطای صفحه[۱۲] کاهش یافته و دسترسی به حافظه به‌طور کلی سریع‌تر می‌شود. در نتیجه زمان‌هایی در حین کار سیستم وجود خواهند داشت که بخشی از حافظه خالی است. BIMS از این بخش خالی برای تسریع حافظه بهره می‌برد.

از طرفی، بیشتر برنامه‌ها از ابتدا حداکثر حافظه‌ی مورد نیاز خود را در اختیار نمی‌گیرند. حتی پس از اینکه به‌مقدار حداکثری برسند نیز اغلب در آن شرایط باقی نمی‌مانند. در نتیجه حتی در حین اجرای بزرگ‌ترین برنامه از نظر حجم حافظه‌ی مورد نیاز، باز هم بخش‌هایی از حافظه وجود دارند که مورد استفاده واقع نمی‌شوند. به منظور درک بهتر این موضوع، شکل ۱ الگوی استفاده از حافظه را برای سه برنامه نشان می‌دهد: برنامه‌ی cholesky از مجموعه‌ی SPLASH [۱۱] و برنامه‌های bodytrack و vips از مجموعه‌ی PARSEC [۱۲].

در این شکل، محور عمودی نشان می‌دهد که هر برنامه چقدر حافظه اشغال کرده است. این مقدار به حداکثر حافظه‌ی مورد نیاز آن برنامه تقسیم شده است. بخش ابتدایی این محور در این شکل نشان داده نشده است تا جزئیات بخش‌های بالائی بهتر نمایان شود. محور افقی نیز میزان پیشرفت اجرا را به صورت تعداد دستورات اجراشده نشان می‌دهد. این محور به حداکثر دستورات اجرایی هر برنامه تعدیل‌شده است. همان‌طور که می‌بینید، هیچ‌کدام از این برنامه‌ها، در ابتدای اجرا، تمامی حافظه‌ی مورد نیاز خود را اشغال نمی‌کنند بلکه در حین اجرا میزان حجم حافظه‌ای که در اختیارشان قرار داده شده است تغییر می‌کند. برنامه‌ی Cholesky بعد از طی‌کردن ۵۵٪ از اجرا، به حداکثر حافظه‌ی مورد نیاز خود می‌رسد، vips بعد از اجرای دوسوم از دستورات به ۹۸٪ از حداکثر حافظه می‌رسد و bodytrack نیز کمتر از یک درصد از زمان اجرا را در وضعیت حداکثری از لحاظ حجم حافظه اشغالی سپری می‌کند.

این دو مورد اشاره شده این فرصت را به‌وجود می‌آورند که معماری‌های بهینه‌تری را برای حافظه ارائه کرد. روش BIMS فضاهای خالی حافظه را به‌صورت SLC مورد استفاده قرار می‌دهد و با استفاده از این فضاها یک لایه‌ی میانی برای تسریع دسترسی به سیستم ایجاد می‌کند به این ترتیب که داده‌هایی که به‌تازگی مورد دسترسی قرار گرفته‌اند را از بخش‌های MLC به این بخش‌ها که سریع‌تر هستند منتقل می‌کند.



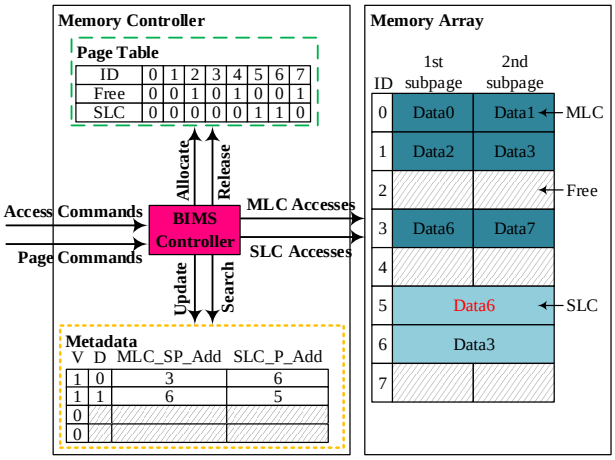
شکل ۱: الگوی استفاده از حافظه برای سه برنامه محک. محور افقی به حداکثر تعداد دستورات اجرایی هر برنامه تعدیل‌شده و محور عمودی به حداکثر میزان حافظه‌ی اشغالی هر برنامه.

۳ -ساختار پیشنهادی BIMS

در این بخش ساختار BIMS ارائه خواهد شد. از آنجایی که مدیریت حافظه‌ی اصلی همواره در سطح صفحه[۱۳] انجام می‌شود، در این روش نیز اندازه‌ی ریزدانگی[۱۴] همان صفحه است. ابتدا، اجزای این معماری معرفی خواهد شد. سپس، مدیریت صفحه‌ی داخلی این روش توضیح داده می‌شود. درنهایت، این روش با ساختارهای متداول حافظه نهان مقایسه خواهد شد.

۳ ـ۱ ساختار BIMS

شکل ۲ ساختار داخلی BIMS را به همراه مثالی از نحوه‌ی به‌کارگیری آن نشان می‌دهد. BIMS دارای چهار جزء اصلی است: آرایه حافظه[۱۵]، جدول صفحه[۱۶]، جدول متاداده[۱۷] و کنترلر[۱۸] .



شکل ۲: ساختار داخلی BIMS. بخش قرمز رنگ (Data6 در SLC) نشان‌دهنده‌ی داده‌ی تغییر یافته است.

آرایه حافظه محل ذخیره‌سازی داده‌هاست. به‌صورت پیش فرض تمامی صفحات موجود در این بخش MLC هستند (یعنی تمامی داده‌های هرکدام از صفحات در سلول‌های MLC ذخیره می‌شوند). اما BIMS سلول‌های تعدادی از این صفحات را به شکل SLC برای افزایش سرعت سیستم به‌کار خواهد گرفت. با فرض اینکه هر سلول MLC می‌تواند m بیت در خود نگه دارد، هر صفحه‌ی فیزیکی که سلول‌های آن MLC باشند، می‌توانند m برابر صفحات فیزیکی که سلول‌های آن‌ها در حالت SLC هستند داده در خود ذخیره کنند. برای مثال، همان‌طور که در شکل ۲ می‌بینید، به ازای m برابر با ۲، هر صفحه‌ی MLC دارای دو زیرصفحه (دو نیمه) است که هرکدام را می‌توان در یک صفحه‌ی SLC ذخیره کرد. در زمان اجرا، اگر BIMS بخواهد از یک صفحه‌ی فیزیکی به‌صورت SLC استفاده کند، می‌تواند هرکدام از این نیمه‌ها را دریکی از صفحات SLC ذخیره کند. در اینجا، فرض می‌کنیم اندازه‌ی هر صفحه‌ی مجازی حافظه ۴ کیلوبایت است و به‌صورت پیش‌فرض نیز به‌صورت MLC ذخیره می‌شود. آنچه سیستم از حافظه می‌بیند نیز این است که همه‌ی صفحات به‌صورت MLC ذخیره می‌شوند تا از حداکثر حجمی که سلول‌ها می‌توانند داشته باشند، استفاده شده باشد.

در روش‌های متداول مدیریت حافظه، هر صفحه در یکی از دو حالت آزاد یا اشغال‌شده قرار دارد و تمایز مشخصی بین صفحات فیزیکی و مجازی وجود ندارد. در BIMS هر صفحه‌ی فیزیکی دارای یکی از سه حالت ممکن است: آزاد، ذخیره‌سازی عالی (اشغال‌شده) و SLC. در نتیجه، یک جدول صفحه وجود دارد که وضعیت هر صفحه را با اختصاص دو بیت به‌ازای هر صفحه تعیین می‌کند. در شکل ۲ این دو بیت بانام‌های "SLC" و "Free" مشخص شده‌اند. اگر صفحه در وضعیت آزاد باشد، بیت Free برابر با ۱ قرار داده می‌شود و بیت SLC را صفر. اگر یک صفحه در حال ذخیره‌سازی داده به‌صورت MLC باشد، هر دو بیت صفر در نظر گرفته می‌شوند و اگر یک صفحه توسط BIMS به صورت SLC مورد استفاده قرارگرفته باشد، بیت SLC برابر با یک خواهد بود و بیت Free برابر صفر.

سربار مساحت بخش حافظه یک جدول صفحه با فرض آنکه از سلول‌های SRAM شش ترانزیستوری ساخته شده باشد، در مقایسه با یک حافظه PCM که از n تا صفحه N کیلوبایتی تشکیل شده است برابر

$$\frac{2 \times 6 \times n}{N \times n} \qquad (۱)$$

است، که در این حالت، زمانی که اندازه هر صفحه ۴ کیلوبایت باشد، این سربار در حدود ۰/۰۳۶٪ خواهد بود. توجه شود که در هر ستون جدول صفحه به دو بیت نیاز داریم، بنابراین تعداد ترانزیستورهای بخش حافظه این جدول ۱۲n می‌باشد.

بخشی به نام جدول متاداده نیز در BIMS وجود دارد. یک آرایه از این جدول، آدرس یک صفحه‌ی SLC را به‌همراه آدرس نیمه صفحه‌ی MLC که داده آن در این SLC ذخیره شده است را در خود نگاه می‌دارد. در شکل ۲ این دو آدرس به‌ترتیب به نام‌های SLC_P_Add و MLC_SP_Add مشخص‌شده‌اند. همچنین به‌ازای هر درایه، یک بیت به نام "V" وجود دارد که یک بودن آن نشان می‌دهد آن درایه معتبر است و یک صفحه‌ی SLC را به یک‌نیمه صفحه‌ی MLC مرتبط می‌کند. یک بیت دیگر نیز برای دنبال‌کردن تغییراتی که ممکن است در صفحه اعمال شده باشد نیز با نام "D" در این جدول وجود دارد. اگر این بیت یک باشد به این معنی است که در صفحه SLC مورد نظر داده‌ای نوشته شده است و لازم است که هنگام آزادسازی این صفحه، داده‌های آن به نیمه صفحه‌ی MLC بازگردانده شود.

نحوه‌ی استفاده از جدول متاداده این‌گونه است که به‌ازای هر دسترسی به حافظه، BIMS بازه‌ی آدرس‌های موجود در ستون SLC_P_Add را با آدرس درخواست داده‌شده، مقایسه می‌کند. اگر آدرس درخواست در بازه‌ای باشد که یکی از SLC_P_Addها بتواند به آن پاسخ دهد، آن درخواست توسط صفحه‌ی SLC مربوطه پاسخ داده می‌شود. از آنجایی که دسترسی به صفحات SLC بسیار بهتر از دسترسی به صفحات MLC است، هرچه اندازه‌ی جدول را بزرگ‌تر کنیم، احتمال استفاده از صفحات SLC بیشتر می‌شود و در نتیجه سیستم بهتر عمل می‌کند. به ازای m برابر با ۲، هر صفحه می‌تواند به دو قسمت تقسیم شود و دو برابر فضا اشغال کند، در نتیجه در بهترین حالت دوسوم کل سلول‌های حافظه را می‌توان به‌صورت SLC استفاده کرد. در نتیجه، برای پوشش این حالت، سربار بخش حافظه این جدول در مقایسه با حافظه PCM برابر

$$\frac{6 \times 1/3 \times n \times (2 + 2\lceil \log (n \times N) \rceil)}{N \times n} \qquad (۲)$$

می‌باشد، که برای اندازه صفحه ۴ کیلوبایت، حداکثر سربار کمتر از ۰/۲۵٪ می‌باشد. توجه شود که تعداد بیت موجود در هر سطر این

جدول
$2 + 2\lceil \log (n \times N) \rceil$ است.

همانند ساختارهای متداول حافظه‌های نهان، این جدول به‌صورت مجموعه‌های انجمنی[۱۹] نگه‌داری می‌شوند تا هم سرعت جست‌وجو بیشتر شود و هم از فضای اختصاص داده‌شده به جدول به‌صورت بهینه استفاده شود. به‌منظور پویا نگاه‌داشتن تعداد صفحاتی که می‌توانند به این صورت نگاه‌داری شوند، تعداد مجموعه‌های موجود را تغییر می‌دهیم اما انجمنی هر مجموعه همواره ثابت است. همانند حافظه‌ی نهان متداول، مجموعه احتمالی مربوطه با استفاده از آدرس درخواست ورودی تعیین می‌شود و برای یافتن درایه‌ی موجود، آدرس‌های SLC_P_Add جست‌وجو می‌شوند. این جدول در کنترل‌کننده حافظه قرار دارد و باید با استفاده از فناوری‌های سریع‌تر از PCM ذخیره شود تا فرایند جست‌وجو تسریع شود.

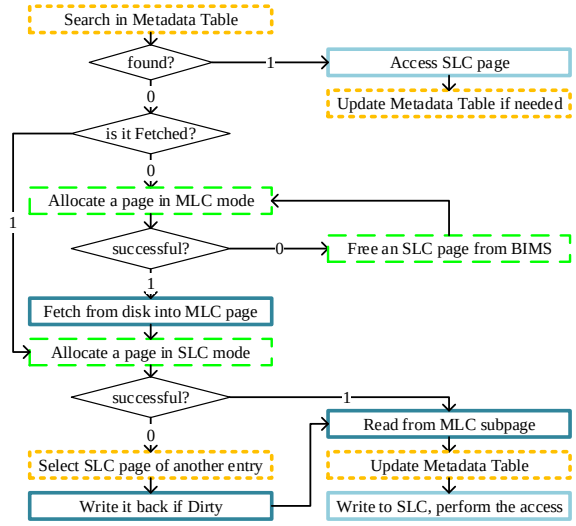
آخرین بخش نیز کنترل‌کننده است که وظیفه‌ی دریافت درخواست‌های ورودی (نوشتن، خواندن و کنترل صفحه) و انجام هرکدام را به‌عهده دارد. تنها این بخش به آرایه حافظه دسترسی مستقیم دارد. همچنین وظیفه‌ی اداره‌کردن جدول صفحه و جدول متاداده نیز به عهده‌ی همین بخش است. تمامی وظایف مدیریتی که در بخش‌های بعدی بررسی می‌شود را کنترل‌کننده اجرا می‌کند.

- **روند دسترسی**[۲۰]

هنگام شروع به کار سیستم، کنترل‌کننده موجود در BIMS، تمامی صفحات را به‌صورت MLC درمی‌آورد به این ترتیب که تمامی درایه‌های موجود در جدول متاداده را نامعتبر می‌کند. با رسیدن درخواست‌ها به حافظه، BIMS به‌تدریج با پرکردن جدول، نیمه صفحات MLC را به صفحات SLC مرتبط می‌کند.

شکل ۳ جریان هر دسترسی به حافظه را نشان می‌دهد که توسط کنترل‌کننده استفاده می‌شود. خط‌چین‌های دور هر بخش نشان می‌دهد که کدام قسمت از شکل ۲ (با خط‌چین‌های شبیه) برای اجرای آن درگیر خواهد بود.

به‌ازای هر درخواست، ابتدا کنترل‌کننده یک مجموعه را از جدول متاداده بر اساس آدرس درخواست ورودی انتخاب می‌کند. سپس بین درایه‌های معتبر در ستون MLC_SP_Add دنبال موردی می‌گردد که شامل این آدرس باشد. در صورتی که چنین مورد یافت شود به این معنی است که نیمه صفحه‌ی مربوطه به‌صورت SLC در نقطه‌ای از حافظه ذخیره شده که آدرس آن در SLC_P_Add درایه‌ی مربوطه ذخیره شده است. در نتیجه کافی است برای پاسخ به درخواست ورودی این صفحه مورد دسترسی قرار گیرد. در صورتی که درخواست ورودی از نوع نوشتن باشد، کنترل‌کننده موظف است بیت D از این درایه را به یک تغییر دهد.



شکل ۳: روند دسترسی داده در BIMS.

اگر جست‌وجو موفق نباشد، به این معنی است که نیم صفحه‌ی مورد نظر به یک صفحه‌ی SLC نگاشته نشده است. در این حالت، کنترل‌کننده یک‌بار جدول صفحه را بررسی می‌کند تا مطمئن باشد که صفحه‌ی مورد نظر در حافظه وجود دارد. اگر چنین باشد، یک بار دیگر جدول صفحه را جست‌وجو می‌کند تا یک صفحه‌ی فیزیکی خالی پیدا کند. اگر صفحه‌ی خالی یافت شود، نیمه صفحه‌ی مربوطه از صفحه‌ی MLC خوانده شده و در این صفحه به‌صورت SLC ذخیره می‌شود و بیت V در جدول صفحه، برای این صفحه یک می‌شود. یک درایه در جدول متاداده نیز به این عمل اختصاص داده می‌شود که در آن SLC_P_Add آدرس صفحه‌ی SLC را نشان می‌دهد و MLC_SP_Add آن آدرسی نیم صفحه‌ی MLC. تمامی درخواست‌های آتی به این نیم صفحه، از طریق صفحه‌ی SLC که به آن اختصاص داده شده است پاسخ داده می‌شوند.

اگر صفحه‌ی خالی پیدا نشود، کنترل‌کننده یکی از صفحات SLC موجود را برای این کار انتخاب می‌کند. این صفحه از میان صفحات موجود در همان مجموعه انتخاب می‌شود و اولویت با صفحاتی است که بیت D مربوطه برایشان صفر باشد تا نیازی به بازنویسی در MLC نداشته باشند. سپس، درایه‌ی مربوطه در جدول متاداده به‌روز می‌شود و داده از نیم صفحه‌ی MLC به صفحه‌ی SLC مورد اشاره در جدول منتقل می‌شود.

اگر آدرس مربوطه در جدول صفحه موجود نباشد، ابتدا صفحه‌ی هدف را از حافظه‌ی جانبی در حافظه‌ی اصلی به‌صورت MLC نوشته و بعد کارهای لازم را مطابق با آنچه در بالا توصیف شد، توسط کنترل‌کننده انجام خواهد شد. در صورتی که صفحه‌ی MLC خالی برای نوشتن داده از حافظه‌ی جانبی وجود نداشته باشد، لازم است یکی از صفحات SLC که مورد استفاده‌ی BIMS است برای این منظور آزاد شود. در اینجا، این صفحه به‌صورت تصادفی از میان صفحات SLC موجود انتخاب می‌شود. روشن است که اگر بیت D مربوط به آن صفحه‌ی باشد، لازم است داده‌های موجود در آن در نیم‌صفحه‌ی

مربوطه بازنویسی شده باشد پیش از آنکه داده از حافظه‌ی جانبی در آن نوشته شود.

به‌عنوان مثالی از نحوه‌ی عملکرد این روش، شکل ۲ حافظه‌ای را نشان می‌دهد که تنها هشت صفحه‌ی فیزیکی دارد. فرض کنید دو صفحه‌ی اول و صفحه‌ی چهارم از پیش اشغال شده‌اند. جدول صفحه نیز به‌طور متناسب تغییر پیدا کرده است و جدول متاداده نیز یک مجموعه با چهار عضو دارد که همگی نامعتبر هستند.

اولین دسترسی به این صفحه به نیمه‌ی دوم صفحه‌ی دوم است که در شکل ۲ با "Data3" نشان داده شده است. از آنجایی که تمامی درایه‌های جدول متاداده نامعتبرند، این نیم صفحه به‌صورت SLC ذخیره نشده است. برای انجام این کار، BIMS به یک صفحه‌ی آزاد نیاز دارد که با توجه به جدول صفحه چنین صفحه‌ای موجود است و یک درایه‌ی نامعتبر از جدول متاداده. کنترل‌کننده مورد نخست را انتخاب کرده و بیت V آن را یک می‌کند. بخش MLC_SP_Add این درایه از جدول با آدرس نیم‌صفحه‌ی مبدأ (در این مثال نیم‌صفحه‌ی شماره‌ی ۳) پر می‌شود. یک صفحه‌ی آزاد (صفحه‌ی ۶ در این مثال) به‌صورت SLC تخصیص داده می‌شود و آدرس آن در بخش SLC_P_Add نوشته می‌شود. جدول صفحه نیز توسط کنترل‌کننده به‌روز می‌شود. سپس نیم‌صفحه‌ی MLC خوانده شده و در این صفحه ذخیره می‌شود. از آنجایی که هر سلول در این صفحه، تنها یک بیت ذخیره می‌کند، در نتیجه این حجم داده تمامی صفحه‌ی فیزیکی را پر می‌کند. در حین این انتقال، درخواست خواندن نیز پاسخ داده می‌شود. سایر درایه‌های جداول نیز به‌طور مناسب به‌روز می‌شوند.

درخواست دوم یک دستور نوشتن در نیمه‌ی اول صفحه‌ی چهارم است که با "Data6" در شکل ۲ نشان داده شده است. بیشتر مراحل پاسخ‌گویی به این درخواست همانند مورد پیش است: درایه‌ی دوم از جدول متاداده و صفحه‌ی پنجم در جدول صفحه انتخاب می‌شوند و چون درخواست یک دستور نوشتن است، بیت D به یک تغییر می‌یابد.

BIMS درخواست‌هایی که در آینده برای هر یک از این دو نیم‌صفحه به حافظه می‌رسند را با استفاده از صفحات SLC مربوطه پاسخ می‌دهد. به این ترتیب این درخواست‌ها با سرعت بیشتر و انرژی کمتری پاسخ داده می‌شوند. درخواست‌هایی که به این دو نیم‌صفحه وارد نمی‌شوند همانند دو مثال بالا پاسخ داده می‌شوند.

- **مدیریت صفحه**

BIMS به یک مکانیزم مدیریت صفحه‌ی داخلی نیز نیاز دارد. مکانیزمی که صفحات را تخصیص داده و آزاد می‌کند. دو دسته عملیات پایه‌ای برای مدیریت صفحه وجود دارد که توسط کنترل‌کننده اجرا می‌شوند: تخصیص صفحه به‌صورت SLC و آزادسازی آن.

تنها راه افزایش تعداد صفحات SLC این است که این صفحات به‌صورت صریح توسط کنترل‌کننده تخصیص داده شوند. همان‌طور که پیش از این توضیح داده شد، این عمل تنها زمانی لازم می‌شود که آدرس درخواست رسیده به حافظه توسط صفحات SLC موجود پوشش داده نشده باشد، یک درایه‌ی خالی در جدول متاداده یافت شود و فضای خالی در حافظه برای تخصیص یک صفحه به‌صورت SLC وجود داشته باشد. اگر این سه شرط برقرار باشند، یک صفحه‌ی خالی در حافظه به‌صورت SLC تخصیص داده می‌شود و آدرس آن صفحه در درایه‌ی موجود در جدول متاداده درج می‌شود.

به‌طور پیش‌فرض، کنترل‌کننده صفحات SLC را آزاد نمی‌کند. به این معنی که تنها در صورت لزوم و نیاز سیستم این عمل انجام می‌شود. با توجه به توضیحات بخش پیش، در صورتی که نیاز سیستم به حافظه بیشتر شود، تعدادی از این صفحات توسط کنترل‌کننده آزاد شده و به‌منظور ذخیره‌سازی داده‌ای کد در اختیار سیستم قرار داده می‌شوند. تخصیص و آزادسازی صفحات SLC منجر به پویایی حجم داده‌ای است که BIMS دنبال می‌کند. نتایج عملی حاصل از این امر را به‌طور دقیق‌تر در بخش‌های آینده بررسی خواهیم کرد.

۳ ۴ -مقایسه با حافظه‌ی نهان متداول

همانگونه که توضیح داده شد، هدف اصلی ساختار BIMS برای مدیریت حجم حافظه در حافظه‌های PCM می‌باشد. ولی شاید از نظر قابلیت‌ها و نحوه مدیریت دسترسی‌ها، به اشتباه همانند حافظه نهان در نظر گرفته شود، در حالی که با هم اختلاف‌های مشخصی دارند که در ادامه به آن‌ها اشاره خواهد شد.

۱. اندازه‌ی BIMS (تعداد صفحاتی که به‌صورت SLC هستند) پویاست. این اندازه توسط کنترل‌کننده موجود در این روش کنترل می‌شود. این در حالی است که حجم داده‌ای که حافظه‌ی نهان هر سیستم می‌توانند ذخیره کند هنگام طراحی پردازنده مشخص می‌شود و قابل دست کاری نیست.

۲. ساختار متاداده در این دو روش متفاوت است. در BIMS آدرس صفحه‌ی SLC نیز ذخیره می‌شود در حالی که در حافظه‌های نهان چنین آدرسی وجود ندارد چرا که خود داده به حافظه‌ی نهان منتقل می‌شود.

۳. اگر یک مجموعه در جدول متاداده پر شود، دیگر نمی‌توان به آن مجموعه یک صفحه‌ی جدید اختصاص داد. در نتیجه ممکن است حتی باوجود BIMS دسترسی مستقیم به صفحات MLC لازم باشد. درحالی‌که در حافظه‌های نهان، همواره هر آدرس حداقل به یک بلوک داده نگاشته می‌شود.

۴ نتایج

۴ ـ ۱ محیط شبیه سازی

به‌منظور ارزیابی BIMS از شبیه‌سازی مبتنی بر اثر[۲۱] استفاده شده است. ابتدا اثرهای مربوط به برنامه‌های مختلف با اجرای هرکدام، استخراج شده و سپس، به‌عنوان ورودی به یک شبیه‌ساز حافظه داده شده است. طراحی و پیاده‌سازی این شبیه‌ساز توسط نویسندگان این اثر انجام شده است. در نهایت نیز آمار به‌دست‌آمده از این شبیه‌ساز برای ارزیابی استفاده شده است. در ادامه هرکدام از این مراحل بیشتر بررسی خواهند شد.

با دست‌کاری بخش حافظه‌ی شبیه‌سازMulti2sim [۱۱] توانستیم اثرهای مورد نیاز را تولید کنیم. با مقداردهی پارامترهای مناسب، توانستیم یک هسته‌ی x86 را در مد عملیاتی[۲۲] شبیه‌سازی کنیم که منجر به تولید اثر تمامی حافظه بدون دخالت حافظه‌ی نهان می‌شود. علاوه بر دستورات عادی دسترسی به حافظه (نوشتن، خواندن و مقداردهی اولیه) Multi2Sim را طوری تغییر دادیم که دستورات مدیریت حافظه را نیز ثبت کند. این دستورات شامل تخصیص صفحه و آزادسازی صفحه هستند که در ادامه در شبیه‌ساز حافظه طراحی‌شده به‌منظور پیاده‌سازی یک روش مدیریت حافظه به‌کار رفته است.

فهرست برنامه‌های شبیه‌سازی‌شده در جدول ۱ آمده است. این مجموعه شامل bodytrack، ferret و vips از مجموعه برنامه‌های محک PARSEC و برنامه‌های cholesky، lu، ocean، radiosity و raytrace از مجموعه‌ی SPLASH است. هر یک از این برنامه‌ها با استفاده از یک نسخه‌ی تغییریافته‌ی Multi2Sim که در بالا توضیح داده شد اجرا شده و اثر حاصل برای شبیه‌سازی حافظه در مرحله‌ی بعد استفاده شد.

جدول ۱: فهرست برنامه‌های محک شبیه‌سازی‌شده و خواص دسترسی به حافظه هرکدام.

نام برنامه	میانگین تعداد نوشتن در هر دستور	میانگین تعداد خواندن در هر دستور
Bodytrack	۰/۱۵	۰/۴۳
Ferret	۰/۱۸	۰/۳۵
vips	۰/۱۷	۰/۳۴
cholesky	۰/۱۳	۰/۴۲
Lu	۰/۱۷	۰/۳۳
ocean	۰/۱۱	۰/۴۲
radiosity	۰/۲۱	۰/۳۵
raytrace	۰/۱۵	۰/۳۶

برای اجرای شبیه‌سازی نهایی، یک شبیه‌ساز حافظه به زبان SystemC طراحی و پیاده‌سازی کردیم. این شبیه‌ساز شامل سه بخش قابل پیکربندی است: پردازنده، حافظه‌ی نهان و حافظه‌ی اصلی. بخش پردازنده‌ی این شبیه‌ساز وظیفه‌ی مدل‌کردن یک پردازنده را به پایه‌ای‌ترین شکل ممکن برعهده دارد و این کار را با تفسیر فایل‌های اثر تولیدشده در مرحله‌ی پیش و تولید درخواست‌های مناسب برای حافظه انجام می‌دهد. بخش‌های حافظه‌ی نهان نیز همان‌طور که از نامش پیداست حافظه‌ی نهان را مدل می‌کند. اما بخش حافظه‌ی اصلی وظیفه‌ی پیاده‌سازی ساختار پیشنهادی BIMS و مدل کردن رفتار آن را بر عهده دارد. تعدادی از پارامترهای مورد استفاده در این روش‌ها در جدول ۲ ارائه شده‌اند. علاوه بر این پارامترها، برای جست‌وجو در جدول متاداده به‌اندازه‌ی ۲۰ کلاک ساعت در نظر گرفته شده است. تأخیر دسترسی به حافظه‌ی SLC نیز در حالت خواندن ۸۰ و در حالت نوشتن ۳۰۰ کلاک ساعت است. همچنین لازم به ذکر است که هر سلول MLC در این شبیه‌سازی‌ها دو بیت را در خود ذخیره می‌کند.

جدول ۲: پیکربندی شبیه‌سازی.

نام ماژول	پارامترها
Processor	single x86 core, single-issue, in-order
L1 Cache	128K, private, write-back, 4-way, 64B line, 2 cycles access
L2 Cache	1MB, private, write-back, 8-way, 64B line, 6 cycles access
Main Memory	MLC PCM, 1GB, 160 cycles read, 1000 cycles write, 4KB Page

با استفاده از شبیه‌ساز پیاده‌سازی شده، سیستم به‌ازای هرکدام از برنامه‌ها شبیه‌سازی شد و در انتها، جزئیات اجرای شبیه‌سازی توسط شبیه‌ساز گزارش شد. این گزارش‌ها برای بررسی کارایی BIMS استفاده شد و در بخش بعد مورد بررسی قرار خواهند گرفت.

۴ ـ ۲ محیط شبیه سازی

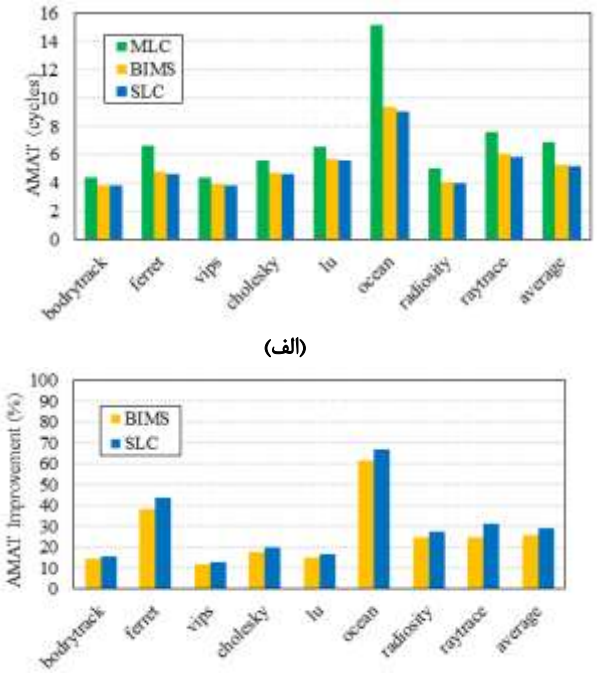
در این بخش نتایج شبیه‌سازی را ارائه می‌کنیم. در هر زیر بخش یکی از پارامترهای طراحی مورد بررسی قرار خواهد گرفت.

- **کارایی**

شکل ۴ (الف) میانگین زمان دسترسی به حافظه[۲۳] را به‌ازای برنامه‌های مختلف نشان می‌دهد. به‌منظور اندازه‌گیری اثر BIMS روی کل سیستم، مقادیر گزارش‌شده شامل زمان دسترسی کل سیستم (شامل حافظه‌های نهان سطح اول و دوم) می‌باشد.

دستگاه‌های مورد مقایسه سه مورد هستند: حافظه‌ی GB۱ که تمام سلول‌هایش MLC هستند، حافظه‌ی GB۱ که تمام سلول‌هایش SLC هستند و حافظه‌ی پیشنهادی BIMS با حداکثر ظرفیت GB۱. همان‌طور که نتایج نشان می‌دهند، به‌ازای تمامی برنامه‌ها، زمان دسترسی به حافظه برای دستگاهی تماماً MLC بیشتر از دو سیستم دیگر است و زمان دسترسی حداقلی به سیستم تماماً SLC تعلق دارد. در این بین زمان دسترسی سیستم مجهز به BIMS، بسیار نزدیک به سیستم تماماً SLC است، گرچه BIMS داده‌ها را به‌طور پیش‌فرض به‌صورت MLC ذخیره می‌کند. به‌طور متوسط زمان دسترسی دستگاهی که تماماً MLC باشد برابر است با ۶/۹ کلاک ساعت در حالی

که این عدد برای سیستم تماماً SLC و سیستم مجهز به BIMS به‌ترتیب برابر است با ۵/۲ و ۵/۳ در مورد مساحت استفاده‌شده، سیستم تماماً SLC دو برابر دو سیستم دیگر مساحت مصرفی دارد، زیرا آن دو سیستم از سلول‌هایی با ظرفیت دو بیت استفاده می‌کنند.



(الف)

(ب)

شکل ۴: مقایسه متوسط زمان دسترسی.

شکل ۴ (ب) میزان بهبود زمان دسترسی برای سیستم مجهز به BIMS و سیستم تماماً SLC را نسبت به سیستم تماماً MLC نشان می‌دهد. سیستم مجهز به BIMS به‌طور متوسط ۲۶٪ و سیستم تماماً SLC به‌طور متوسط ۲۹٪ بهتر از سیستم تماماً MLC عمل می‌کند. درواقع سیستم مجهز به BIMS به سربار مساحت ۰/۷۵٪ کارایی نزدیک به سیستم SLC دارد.

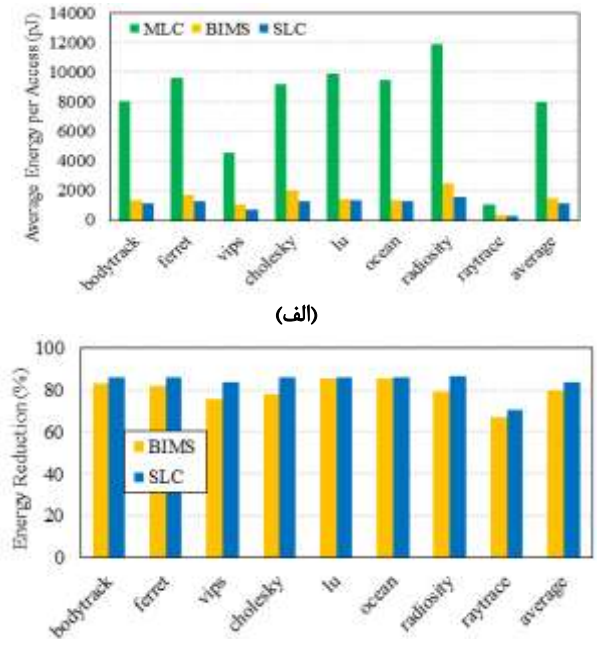
- **انرژی**

تبدیل دسترسی‌هایی که به‌طور پیش‌فرض باید برای اجرای آن‌ها به سلول‌های MLC مراجعه می‌شد به دسترسی‌هایی که تنها به سلول‌های SLC نیاز دارند، باعث کاهش انرژی مصرفی می‌شود. به‌منظور نشان‌دادن این موضوع، از پارامترهای موجود در جدول ۳ برای تخمین انرژی استفاده کردیم. اعداد درج‌شده در جدول ۳ بر اساس کارهای [۱۴، ۱۵، ۱۶] انتخاب شده‌اند و میزان متوسط انرژی مورد نیاز برای خواندن از و نوشتن در هرکدام از فناوری‌های آورده‌شده را نشان می‌دهند. نوشتن ۰۰ و ۱۱ در MLC به‌اندازه‌ی نوشتن ۰ و ۱ در SLC انرژی مصرف می‌کنند. خواندن از سلول‌های MLC نیز در دو مرحله انجام می‌شود، در نتیجه انرژی مصرفی آن دو برابر است. لازم به ذکر است که در این تخمین انرژی، تنها حافظه‌ی اصلی مورد نظر بوده و از پردازنده‌ها به‌همراه حافظه‌های نهان آن‌ها در محاسبه انرژی مصرفی صرف‌نظر شده است.

جدول ۳: پارامترهای مورد استفاده در تحلیل انرژی.

عملیات		انرژی (pJ)
نوشتن MLC	00	۳۶
	01	۳۰۷
	10	۵۴۷
	11	۲۰
نوشتن SLC	0	۳۶
	1	۲۰
خواندن MLC		۴
خواندن SLC		۲

شکل ۵ (الف) متوسط انرژی مصرفی سه سیستم را به‌ازای هر دسترسی نشان می‌دهد. همان‌طور که انتظار می‌رفت، بیشترین انرژی مصرفی متعلق به سیستم تماماً MLC است و کمترین انرژی را سیستم تماماً SLC مصرف می‌کند. انرژی مصرفی سیستم مجهز به BIMS نزدیک به سیستم تماماً SLC است. با توجه به شکل ۵ (ب) انرژی مصرفی سیستم تماماً SLC تا ۸۴٪ کمتر از سیستم تماماً MLC است، درحالی‌که این مقدار برای سیستم مجهز به BIMS، در حدود ۸۰٪ می‌باشد.



(الف)

(ب)

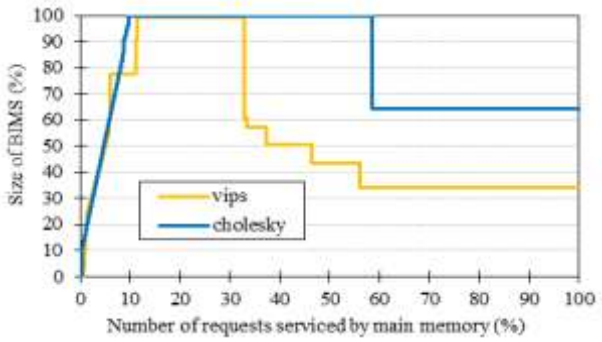
شکل ۵: انرژی مصرفی با استفاده از BIMS.

- **ظرفیت پویای BIMS**

همان‌طور که پیش‌ازاین نیز توضیح داده شد، BIMS تعداد صفحاتی که به‌صورت SLC نگاه می‌دارد را در حین اجرا بر اساس نیاز سیستم و حافظه‌ی موجود تغییر می‌دهد. تعداد این صفحات را حجم BIMS می‌نامیم. اگر آدرس مورد نظر به‌صورت SLC ذخیره نشده باشد و جای

خالی در حافظه موجود باشد، حجم BIMS افزایش می‌یابد. اگر هم حافظه پر باشد و سیستم به حافظه‌ی بیشتری نیاز داشته باشد، تعدادی از صفحات SLC آزاد می‌شوند و حجم BIMS کاهش می‌یابد. اگر هم یک صفحه توسط سیستم آزاد شود (مثلاً سیستم‌عامل یا کاربر بخشی از حافظه را آزاد کند) علاوه بر داده‌هایی که به‌صورت MLC ذخیره شده‌اند، تمامی صفحاتی که بخشی از این داده‌ها را به‌صورت SLC ذخیره کرده‌اند نیز آزاد می‌شوند. در نتیجه به‌طور کلی با این سه روش، حجم BIMS ممکن است افزایش یا کاهش یابد.

به‌منظور نشان‌دادن پویایی اندازه‌ی BIMS، شکل ۶ حجم BIMS را در حین اجرا برای دو برنامه‌ی vips و cholesky نشان می‌دهد. محور عمودی حجم BIMS را در حین اجرا بر اساس حداکثر حجم آن برنامه نشان می‌دهد. محور افقی نیز با نشان‌دادن درصد درخواست‌های از حافظه که پاسخ داده شده‌اند، میزان پیشرفت در اجرای آن برنامه را نشان می‌دهد.



شکل ۶: حجم BIMS در حین اجرا.

در آغاز اجرا، از آنجایی که همه صفحات به‌صورت پیش‌فرض در حالت MLC هستند، حجم BIMS صفر است. با رسیدن درخواست‌ها به حافظه، BIMS تعدادی از صفحات خالی را به حالت SLC در آورده و داده‌هایی که اخیراً مورد استفاده قرار گرفته‌اند را در آن‌ها ذخیره می‌کند و به مرور زمان حجم BIMS افزایش می‌یابد. این فرایند تا جایی ادامه پیدا می‌کند که صفحات SLC تمامی فضای آدرس مورد نیاز برنامه را پوشش داده باشند یا اینکه صفحه‌ی خالی برای تبدیل به SLC وجود نداشته باشد. بعد از مدتی، به‌دلیل تغییر نیاز سیستم، حجم BIMS کم می‌شود. اگر برنامه حافظه‌ی بیشتری نیاز داشته باشد، تعدادی از صفحات SLC آزاد می‌شوند و اگر برنامه حافظه‌ی کمتری بخواهد (حافظه را آزاد کند) صفحات SLC مربوطه نیز توسط BIMS آزاد می‌شوند و به این ترتیب در هر دو حالت حجم BIMS کم می‌شود. این فرایند افزایش و کاهش حجم BIMS ممکن است باز هم رخ دهد. برای نمونه، در مورد برنامه‌ی vips، نخستین بار حجم BIMS پس از اجرای ۳۳٪ درخواست‌ها کم می‌شود. به فاصله‌ی کمی از این امر، حجم BIMS بازهم افزایش یافته و این بار تا نزدیک به پایان اجرای برنامه ثابت می‌ماند. در مورد cholesky، BIMS به مدت طولانی‌تری در حالت حداکثری می‌ماند و تنها یک‌بار دچار کاهش حجم می‌شود. پویایی حجم BIMS کلید اصلی در استفاده از این روش است. به‌دلیل این پویایی، BIMS می‌تواند صفحات خالی را برای افزایش سرعت و کاهش انرژی مصرفی سیستم به‌کار گیرد.

۵ نتیجه‌گیری

در دهه‌ی گذشته، فناوری حافظه‌های PCM به‌عنوان یکی از نامزدهای اصلی جایگزینی حافظه‌های DRAM در حافظه‌های اصلی مطرح شده است. از برتری‌های این فناوری، تراکم بیشتر و انرژی ایستای کمتر آن است. همچنین سلول‌های PCM توانایی ذخیره‌سازی چند بیت را دارند که این امر باعث افزایش توان مصرفی و کاهش سرعت و عمر این حافظه‌ها می‌شود. برای کاهش این هزینه‌ها ما روش BIMS را پیشنهاد می‌کنیم که با استفاده از سلول‌هایی که تنها یک بیت در خود ذخیره می‌کنند، سرعت و عمر سلول‌های چندسطحی را افزایش داده و انرژی مصرفی سیستم را کاهش می‌دهد. بر اساس شبیه‌سازی‌های انجام‌شده با استفاده از شبیه‌ساز گسترش داده‌شده توسط خودمان، این روش در مقایسه با حافظه‌ای که تمام سلول‌هایش چندسطحی هستند ولی از این روش استفاده نمی‌کند، انرژی مصرفی و زمان دسترسی را به‌طور متوسط و به‌ترتیب ۸۰٪ و ۲۶٪ کاهش می‌دهد. این دستاوردها تنها با هزینه‌ی نزدیک به ۰/۷۵٪ مساحت به‌دست آمده‌اند.

مراجع

[۱] سعید پارسا و محمد حمزه‌یی، "کاشی‌بندی حلقه‌های تودرتو با در نظر گرفتن محلیت داده‌ها به‌منظور اجرای موازی بر روی پردازنده‌های چندهسته‌ای"، مجله مهندسی برق دانشگاه تبریز، جلد ۵۴، شماره ۳، صفحات ۱۷-۲۶، پاییز ۹۴.

[۲] سعیده نبی‌پور، جواد جاویدان و غلامرضا زارع فتین، " طراحی یک دیکدر BCH بهینه جهت افزایش اطمینان در ذخیره‌سازی اطلاعات و تصحیح خطا در حافظه‌های فلش"، مجله مهندسی برق دانشگاه تبریز، جلد ۴۶، صفحات ۳۱۹-۳۳۱، شماره ۳، پائیز ۹۵.

[3] M.K. Qureshi, D-H Kim, S. Khan, P.J. Nair and O. Mutlu, “AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems”, in Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 1-11, 2015.

[4] S. Ovshinsky, “Reversible Electrical Switching Phenomena in Disordered Structures,” Physical Review Letter, vol. 21, no. 20, pp. 1450–1453, 1968.

[5] M. K. Qureshi, S. Gurumurthi and B. Rajendran, *Phase Change Memory: From Devices to Systems*, Morgan & Claypool Publishers, 2011.

[6] A.L. Lacaita and A. Redaelli, “The race of phase change memories to nanoscale storage and applications”, Microelectronic Engineering, vol. 109, pp. 351-356, 2013.

[7] Y. Choi, I. Song, M.-H. Park, H. Chung, S. Chang, B. Cho, J. Kim, Y. Oh, D. Kwon, J. Sunwoo, J. Shin, Y. Rho, C. Lee, M. G. Kang, J. Lee, Y. Kwon, S. Kim, J. Kim, Y.-J. Lee, Q. Wang, S. Cha, S. Ahn, H. Horii, J. Lee, K. Kim, H. Joo, K. Lee, Y.-T. Lee, J. Yoo and G. Jeong, “A 20nm 1.8V 8Gb PRAM with 40MB/s program bandwidth,” in Proceedings of IEEE International Solid-State Circuits Conference, pp. 46–48, 2012.

[8] M. Han, Y. Han, S.W. Kim, H. Lee and I. Park, “Content-Aware Bit Shuffling for Maximizing PCM Endurance”, ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 22, no. 3, Article no. 48, 2017.

[9] N. Papandreou, H. Pozidis, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam and E. Eleftheriou, "Programming algorithms for multilevel phasechange memory," in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), pp. 329–332, 2011.

[10] O. Zilberberg, S. Weiss and S. Toledo, "Phase-Change Memory: An Architectural Perspective," ACM Computing Surveys, vol. 45, no. 3, Article no. 29, 2013.

[11] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," ACM SIGARCH Computer Architecture News, vol. 23, no. 2, pp. 24–36, 1995.

[12] C. Bienia, S. Kumar, J. P. Singh and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, pp. 72-81, 2008.

[13] R. Ubal, B. Jang, P. Mistry, D. Schaa and D. Kaeli, "Multi2Sim: A Simulation Framework for CPU-GPU Computing," in Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques, pp. 335-344, 2012.

[14] F. Bedeschi, R. Fackenthal, C. Resta, E. M. Donze, M. Jagasivamani, E. C. Buda, F. Pellizzer, D. W. Chow, A. Cabrini, G. M. A. Calvi, R. Faravelli, A. Fantini, G. Torelli, D. Mills, R. Gastaldi and G. Casagrande, "A Bipolar-Selected Phase Change Memory Featuring Multi-Level Cell Storage," IEEE Journal on Solid-State Circuits, vol. 44, no. 1, pp. 217–227, 2009.

[15] J. Wang, X. Dong, G. Sun, D. Niu and Y. Xie, "Energy-efficient multi-level cell phase-change memory system with data encoding," in Proceedings of IEEE 29th International Conference on Computer Design, pp. 175–182, 2011.

[16] F. Bedeschi, C. Resta, O. Khouri, E. Buda, L. Costa, M. Ferraro, F. Pellizzer, F. Ottogalli, A. Pirovano, M. Tosi, R. Bez, R. Gastaldi and G. Casagrande, "An 8Mb demonstrator for high-density 1.8V Phase-Change Memories," in Proceedings of Symposium on VLSI Circuits, pp. 442-445, 2004.

زیرنویس‌ها

[1] Dynamic Random Access Memory
[2] Phase Change Memory
[3] Crystalline
[4] Amorphous
[5] Heater
[6] Refresh
[7] Multi-Level Cell
[8] Single-Level Cell
[9] Program-and-Verify
[10] Built-in Intermediate Memory Structure
[11] Locality
[12] Page Fault
[13] Page
[14] Granularity
[15] Memory Array
[16] Page Table
[17] Metadata Table
[18] Controller
[19] Associative Sets
[20] Access Flow
[21] Trace-Base Simulation
[22] Functional
[23] Average Memory Access Time (AMAT)