The First National Conference on Software Engineering Applications
3-5 march 2009
اولین کنفرانس ملی
کاربردهای مهندسی نرم افزار لاهیجان

تاثیرات تجزیه و تحلیل پیکر بندی حافظه نهان در هنگام پیش بینی و تشخیص انشعابات غیر مستقیم

فاطمه حیدری، سمانه فتاحی ، حسین بیگی

دانشگاه ازاد اسلامی واحد دولت اباد

دانشگاه ازاد اسلامی واحد دولت اباد

دانشگاه ازاد اسلامی واحد دولت اباد

Heidari_6f6@yahoo.com, Hossein_beigi@yahoo.com

چکیده

این مقاله تاثیرات استفاده از حافظه نهان را روی پیش بینی انشعابات غیر مستقیم در پردازنده های ILP مورد بحث قرار می دهدو سهم اصلی ان شناخت این واقعیت است که محتوای حافظه نهان حاوی اطلاعاتی درباره جریان کنترلی برنامه اخیر است که می تواند دقت پیش بینی کننده ها را که خودشان قادر به استفاده صریح از چنین اطلاعاتی نمی باشند ، افزایش دهند. ما نشان می دهیم که آدرس مسیر انشعاب غیر مستقیم در حافظه نهان دقت پیش بینی انشعاب غیر مستقیم را افزایش می دهد . پس بصورت تدریجی فشرده سازی، اضافه کردن شمارنده بیتی در هر خط حافظه نهان شرکت پذیری حافظه نهان، سایز آن و . . . را برای تاثیر هر پیکر بندی روی پیش بینی انشعاب غیر مستقیم مورد بررسی قرار داده ایم. این مقاله تکنیک جدیدی را برای استفاده کافی از trace cache های کوچک ارائه می دهد . یک trace cache می تواند کارایی پردازنده ها را بطور قابل توجهی افزایش دهد. trace cache ها بعنوان راه حلی برای مشاله مکانیسم واکشی در پردازنده های اخیر بطور موثری مورد استفاده قرار گرفته اند.

کلمات کلیدی

trace cache ، integer ، انشعابات غیر مستقیم

۱-مقدمه

پردازشگرهای (ILP) ، دستورالعمل موازی سطح ، خواه superscalar و یا VLIW بر واحد های کاربردی گسترده ای برای استخراج ILP از application ها دارد . موتور اجرای گسترده به واحد fetch (واکشی) دستور العمل بزرگتری نیاز دارد، که باید پهنای واحد کاربردی آن ها به منظور استفاده بهینه از همه آنها با هم match (هماهنگ) باشد .

در واقع واحد واکشی دستورالعمل باید تعداد زیادی از دستورالعمل ها را از cacheمیان بلاک های اصلی در هر سیکل ، واکشی کند . عموما این بلاک های اصلی جایگزین موقعیت های غیر پیوسته در cache می شوند . بیش از یک دسترسی به cache باید به منظور واکشی همه پلاک های اصلی مورد نیاز ، صورت گیرد . مکان های واکشی غیر پیوسته در تعداد دستورالعمل ها محدود است . از این رو cache را همکار مناسبی برای واکشی چندین بلاک اصلی در هر سیکل محسوب می گردد .

Trace cache (اثر کش) تکنیک ریز معماری برای افزایش پهنای باند واکشی دستورالعمل یک پردازنده superscalar می باشد . پردازنده ، جریان دستورالعمل اجرا شده را جمع آوری کرده و آنها را در خطوط Trace cache با آدرس های مسیر انشعاب ذخیره می نماید . در این روش ، دستورالعمل های واکشی شده، از مسیر اجرای غیر پیوسته در موقعیت های پیوسته در Trace cache جایگزین می گردد . هنگامی که یک خط Trace cache به مسیر اجرای بعدی ، ارجاع داده می شود ، خط کلی را می توان در طی یک دسترسی به cache واکشی نمود . [1] . مکانیسم های واکشی و اجرا بوسیله بافرها برای مثال صف ها ، مکانیسمهای رزرو و . . . مجزا شده اند . مکانیسم واکشی دستورالعمل مانند یک تولید کننده ای که عمل واکشی دیکد و غیره را انجام می دهد ، عمل می کند .[2] کارایی یک پردازنده تک thread به شدت وابسته به میانگین تعداد دستورالعمل های مفیدی است که می توان آنها را درسیکل واکشی نمود . trace cache ها دربرآورده سازی این هدف با ذخیره دستورالعمل ها بصورت پویا بسیار مؤثر هستند [3] . trace cache ها بعنوان راه حل موثری برای مشخص کردن محدودیت های مکانیسم واکشی پیشنهاد شده اند . این cache ها بلاک های اصلی دستورالعمل را به روشی که مجدداً در طی اجرای برنامه قابل بازیابی باشند ، ذخیره می کند . هر خط trace یک دنباله از بلاکها را بااطلاعات نزدیک به بلاکها ذخیره می نماید . برای پیش بینی انشعاب بعدی درطی هر واکشی چندین پیش بینی کننده انشعاب مورد استفاده قرار گرفته است . [4] دراین مقاله, ما به ترتیب بخش های زیر را دنبال می کنیم :

2-1-بخش اول

عموماً ، هدف اصلی c و برنامه های Fortran که دارای انشعابات غیر مستقیم کمتر ، انشعابات شرطی و بازگشت های تابعی کمتری می باشند ، این است که انشعابات قابل پیش بینی باشند . اگرچه ، فرکانس استفاده از انشعابات غیر مستقیم بیشتر از کاربردهای آینده است ، اگر چه انشعابات شرطی در بیشتر application ها ، ادامه می یابند . برای مثال برنامه های شیء گرا object oriented ، از انشعابات غیر مستقیم بیشتری برای پیاده سازی فراخوان های تابع مجازی استفاده می نمایند . کتابخانه های نیک شده بصورت دینامیکی (dll) با استفاده از انشعابات غیر مستقیم فراخوانی می شوند . نهایاتا افزایش تعداد زبانهای برنامه نویسی با استفاده از ماشین های مجازی (VMS) نظیر java به تکنیکی نیاز دارند که تعداد زیادی انشعاب غیر مستقیم را دارا باشد . بنابراین نرخ دقت در پردازنده های ILP افزایش می یابد .

ما مشاهده می کنیم که در یک پردازنده ILP با Trace cache، امکان افزایش دقت پیش بینی انشعاب غیر مستقیم برنامه ها با استفاده از Trace cache به جای استفاده از بافر مسیر انشعاب وجود دارد . بعلاوه ، استفاده از این طرح مستلزم ایجاد ساختار بسیار ساده سخت افزاری بدون استفاده از جداول اضافی می باشیم .

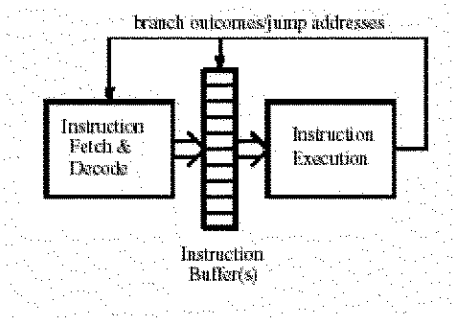
در این مقاله نشان می دهیم چگونه Trace cache می تواند برخی از محتوای اطلاعاتی مورد استفاده بوسیله پیش بینی کننده های غیر مستقیم دوسطحی را تقسیم کند . اگر چه افزایش دقت، کمتر از انشعاب غیر مستقیم دو سطحی می باشد و هزینه نیز کمتر است . اگر کسی قصد ساخت یک Trace cache را داشته باشد ، باید قادر باشد پیش بینی غیر مستقیم را با یک BTB نرخ کم با هزینه غیر اضافی انجام دهد . ما در این مقاله برای مقایسه پیش بینی کننده های انشعاب غیر مسقتیم دو سطحی یا nent trace ناکشی نکرده ایم . ما منحصراً مزایای دو سطحی را با استفاده از Trace cache نشان داده ایم . مبنی بر مشاهدات اولیه ما مکارهایی را برای افزایش دقت ارائه می دهیم ۱) پیشنهاد می کنیم آدرس مسیر انشعاب غیر مستقیم را در Trace cache بهنگام سازیم ما استفاده از سایت بهنگام سازی را پیشنهاد می نمائیم. ۲) نشان می دهیم بهنگام سازی با شمارنده ۲ بیتی در انتهای هر خط انجام گیرد.

2-2-بخش دوم

طبق شکل ۱ ، پردازنده های superscalar با کارایی بالا به مکانیسم واکشی دستورالعمل و مکانیسم اجرای دستورالعمل تقسیم می شود . موتور اجرای دستورالعمل مصرف کننده ای است که دستور العمل ها را از بافر حذف

کرده و سپس آنها را اجرا می کند . وابستگی های کنترلی (انشعاب ها و jump ها) یک مکانیسم باز خورد را میان تولید کننده و مصرف کننده بوجود می آورد .

پردازنده هایی که از این تکنیک ها استفاده می نمایند ، سعی در اجرای دستورالعمل بصورت موازی با هم دارند . بافر های بزرگ برای حفظ یک قالب دستور العملی برای اجرا بصوورت موازی مورد استفاده قرار می گیرند . برای فعال نمودن اجرای همروند دستورات ، موتور اجرایی از واحد های کاربردی زیادی که بصورت موازی با هم قرار گرفته اند عمل می نماید . موتور واکشی انشعاب ها را برای اجرای پیوسته دستورات به درون قالب window ارسال می کند . هدف از این طرح افزایش مقیاس این تکنیک هاست :



شکل ۱ :تجزیه موتورهای عمل گر واکشی

Dispatch وسیعتر ، قابهای بزرگتر ، ثابت های فیزیکی بیشتر، واحد های کاربردی بزرگتر و . . . . برای حصول این اهداف ، برقراری تعادل میان همه قسمتهای پردازنده بسیار مهم است .

در این مقاله ، ما پهنای باند واکشی دستورالعمل را برای افزایش کارایی مورد بررسی قرار داده ایم . کارایی واکشی دستورالعمل وابسته به تعدادی از فاکتورها می باشد . نرخ اصابت و دستورالعمل cacheو دقت درش بینی انشعاب مسائل بسیار مهمی است که در چنین واکشی دستورالعمل مورد استفاده قرار می گیرد . در این مقاله ، فاکتورهای دیگری را که در هر سیکل واکشی مؤثر است بررسی نموده ایم :

بازدهی انشعاب : اگر فقط در هر سیکل یک انشعاب شرطی اجرا شود ، می توان window را در نرخ فقط یک بلاک اصلی در هر سیکل افزایش داد . پیش بینی چندین انشعاب در هر سیکل باعث می شود بازدهی دستورالعمل کلی بالاتر گردد .

تنظیم دستورالعمل غیر پیوسته – بدلیل انشعابات و پرش ها jumpها ، و دستورالعمل ها باید در طی هر سیکل داده شده واکشی گردند . از این رو ، باید مسیر های مناسب و منطقی برای واکشی و تنظیم بلاک های اصلی غیر پیوسته و عبور آنها به گذر گاه وجود داشته باشد . فقط کافی نیست که دستورالعمل ها در cache حضور داشته باشند ، بلکه باید آنها بصورت موازی قابل دسترسی باشند .

دوام واحد واکشی ، دوام گذرگاه تاثیر مهمی روی کارایی پردازنده دارد . این تاثیر ناشی از هزینه پر کردن مجدد گذرگاه بعد ازکنترل غیر صحیح می باشد . درمورد واجد واکشی ، هزینه شروع واکشی مجدد بعد از کم کردن انشعاب ، Jump یا دستورالعمل cache بسیار مهم است . نیاز به بازدهی انشعاب بیشتر و تنظیم دستورالعمل غیر پیوسته دوام واحد واکشی افزایش خواهد یافت .

واحدهای واکش کنونی به یک پیش بینی انشعاب درهرسیکل محدود شده می تواند ۱ بلاک اصلی را در هر سیکل واکشی کند .

داده های موجود درجدول ۱ ، میانگین سایز بلاک های اصلی را که در حدود ۴یا ۵ دستورالعمل برای کدهای صحیح می باشد ، نشان می دهد . واکشی یک بلاک اصلی درهر سیکل برای پیاده سازی بیش از ۴ دستورالعمل درهر سیکل کافی می باشد . اگر ما چندین پیش بینی انشعاب را معرفی می نمائیم ، واحد واکشی در نهایت می تواند چندین بلاک اصلی پیوسته را درهر سیکل واکشی نماید . داده هایی که برای چندین دستورالعمل میان انشعابات گرفته شده مورد استفاده قرارمی گیرند تا حدی به دلیلی فراکانس انشعابات محدود شده است . بنابراین ، اگر یک انشعاب گرفته شود ، واکشی دستورات زیرمسیر قبلی درهمان سیکلی که انشعاب واکشی شده است ، مورد نیاز می باشد .

جدول (۱) : انشعاب و بلاک اصلی

Benchmark	taken %	avg basic block size	# instr between taken branches
eqntott	86.2%	4.20	4.87
espresso	63.8%	4.24	6.65
xlisp	64.5%	4.34	6.70
gcc	62.6%	4.65	6.88
sc	70.2%	4.71	6.71
compress	60.9%	5.39	8.85

2-3-بخش سوم

استفاده از trace cache ، اگر دارای سایز محدود شده ای نباشد ، بسیار مؤثر است . اگر چه trace cache های

کوچک به علت زمان دسترسی و کارایی حافظه بسیار کاربردی تر است : بلاک های اصلی را می توان در trace cache های مختلفی مشاهده نمود ، بعضی از trace ها حاوی ماکزیمم دستورالعمل ها نمی باشد و بلاک مورد در خواست ممکن است درون trace باقی مانده و غیر قابل دسترس گردد . بنابراین کلید برای استفاده مؤثر از trace cache های بسیار کوچک ، ارائه روشی برای تغییر ساخت آنهاست . این مقاله برروی دومین قسمت منطقی مورد نیاز برای ساخت trace ها متمرکز شده است . فیلترینگ هم اکنون روش پیشنهادی برای افزایش تاثیر سایز trace cache میباشد . دراین روش پیشنهاد می شود ، همه trace هایی که حاوی انشعابات هستند ذخیره شوند . پس باید آنها را مبنی بردرصد استفاده نشان فیلتر کرد . trace cache به دو بلاک اصلی تقسیم شده است : (FTC) , (MTC) . همه trace ها روی FTC نوشته می شود . اما فقط trace های موجود در MTC مفید می باشند . موفقیت این روش فیلترینگ مبنی برمشاهده آنهاست ، بیشتر trace های ساخته شده و قرار گرفته در trace cache قبل از eviction مورد استفاده قرارگرفته و بیشتر دستورالعمل ها از مجموعه trace های کوچک اجرا می شود . پیشنهاد شده است که به منظور فیلتر کردن trace هایی که غالباً مورد استفاده قرارگرفته اند ، نمونه برداری شده و لوکالیتی آنها نشان داده شود . این مقاله نوع جدیدی از تکنیکها ی فیلترینگ را که مبنی برنمونه برداری آماری trace ها می باشد معرفی می نماید .این نوع فیلتر ها می خواهد کیفیت trace ها رادریک trace cache کوچک افزایش دهد . این مقاله کارایی و توان فیلتر نمونه برداری اصلی (sf) وورژن ارتقاء یافته آن را به همراه مقایسه آن با ftc -mtc تحلیل وبررسی می کند .

2-4-بخش چهارم

پردازنده های superscalar قادر است تعداد زیادی از دستورالعمل ها را درقالب یک سیکل واحد اجرا نماید . اگر چه مکانیسمهای واکشی دستورالعمل باید دستورالعمل ها را با مکانیزم موازی اجرا نماید . دقت پیش بینی انشعاب ، نرخ اصابت در cache و قواسیهای اجرای دینامیکی فاکتور های تاثیر گذار در کارایی مکانیسم واکشی می باشند .

آدرس بلاک شروع و بیت های آن برای تعیین حضور trace مورد نیاز در trace cache مورد استفاده قرار گرفته است .

اگر trace cache حضور داشته باشد ، آن از trace cache یک سیکل واحد بازیابی شده است . درحالیکه trace cache ها به حصول کارایی بهتر کمک میکنند اما دارای محدودیت هایی نیز می باشند . برخی از این محدودیتها عبارتند از :

- دیده شده است که با استفاده از trace cache های بزرگتر ، مزایای مربوط به کارایی کاهش می یابد . درواقع مصرف توان افزایش می یابد .
- trace هایی که کاملاً شامل بلاک های پیوسته باشند احتمالاً درخطوط آدرس دهی cache موجود می باشند . حضور چنین trace هایی منجر به ایجاد افزونگی و هدر رفتن فضای حافظه خواهد شد .
- با توجه به اینکه ۸۰٪ زمان درزمینه ۲۰ ٪ از کدها صرف می شود ، ممکن است LRU روش بهتری باشد .

راهکار ما برای حل این مسائل استفاده از نموداری است که برای طراحی جایگزینی trace ها در trace cache مورد استفاده قرار می گیرد . هربرنامه کاربردی به پارتیشن های کوچکتر تقسیم می شود و هریک ازاین زیرمجموعه ها نمودار می شوند لوکالیتی درجریان نمونه افزایش می یابد .

کاربرد اصلی این پروسه تعریف و تشخیص trace هایی است که غالباً مورد استفاده قرارگرفته است .این پروسه درواقع برای تقسیم trace ها به دو سطح بصورت زیربسیار مهم هستند هرزمانی که یک trace به دورن trace cache آورده می شود جزئیات آن با trace های موجود درجدول چک می گردد.

اگر اطلاعاتی یافت نشود ، trace به درون cache وارد شده و شمارنده ان را تغییر می یابد . بنابراین هروقت trace مورد اصابت قرار می گیرد ، شمارنده اننیز افزایش می یابد هنگامی که درصد استفاده آن به استانه t1 می رسد ، جزئیات آن به یکم مدخل در cache نوشته می شود . اگر بعد از این trace از cache بازیابی شود ، مدخل جدول به اخرین مقدار بهنگام سازی می گردد . اگر بعد ازاین زمان ، همان trace به cache وارد شود ، این جزئیات به شمارنده انتقال داده می شود.

هنگامی که مقدرا شمارنده به دومین آستانه t2 برسد ، شمارنده بهنگام سازی می گردد . توجه کنید که شمارنده استفاده از آنها دارای مقادیر بزرگتر از t2 می باشد ، قادر به حذف شدن نمی باشند . اگر جدول با چنین مداخلی پرشود ، ماشروع به درخواست برای هر مدخل می کنیم . هنگامی که شمارنده به ۱۵ برسد ، پروازنده متوقف شده و

The First National Conference on Software Engineering Applications
3-5 march 2009
اولین کنفرانس ملی
کاربردهای مهندسی نرم افزار لاهیجان

تاثیرات تجزیه و تحلیل پیکر بندی حافظه نهان در هنگام پیش بینی و تشخیص انشعابات غیر مستقیم

فاطمه حیدری، سمانه فتاحی ، حسین بیگی
دانشگاه ازاد اسلامی واحد دولت اباد
دانشگاه ازاد اسلامی واحد دولت اباد
دانشگاه ازاد اسلامی واحد دولت اباد

Heidari_6f6@yahoo.com, Hossein_beigi@yahoo.com

چکیده

این مقاله تاثیرات استفاده از حافظه نهان را روی پیش بینی انشعابات غیر مستقیم در پردازنده های ILP مورد بحث قرار می دهدو سهم اصلی ان شناخت این واقعیت است که محتوای حافظه نهان حاوی اطلاعاتی درباره جریان کنترلی برنامه اخیر است که می تواند دقت پیش بینی کننده ها را که خودشان قادر به استفاده صریح از چنین اطلاعاتی نمی باشند ، افزایش دهند. ما نشان می دهیم که آدرس مسیر انشعاب غیر مستقیم در حافظه نهان دقت پیش بینی انشعاب غیر مستقیم را افزایش می دهد . پس بصورت تدریجی فشرده سازی، اضافه کردن شمارنده بیتی در هر خط حافظه نهان شرکت پذیری حافظه نهان، سایز آن و . . . را برای تاثیر هر پیکر بندی روی پیش بینی انشعاب غیر مستقیم مورد بررسی قرار داده ایم. این مقاله تکنیک جدیدی را برای استفاده کافی از trace cache های کوچک ارائه می دهد . یک trace cache می تواند کارایی پردازنده ها را بطور قابل توجهی افزایش دهد. trace cache ها بعنوان راه حلی برای مساله مکانیسم واکشی در پردازنده های اخیر بطور موثری مورد استفاده قرار گرفته اند.

کلمات کلیدی

trace cache ، integer ، انشعابات غیر مستقیم

۱-مقدمه

پردازشگرهای (ILP) ، دستورالعمل موازی سطح ، خواه superscalar و یا VLIW بر واحد های کاربردی گسترده ای برای استخراج ILP از application ها دارد . موتور اجرای گسترده به واحد fetch (واکشی) دستور العمل بزرگتری نیاز دارد، که باید پهنای واحد کاربردی آن ها به منظور استفاده بهینه از همه آنها با هم match (هماهنگ) باشد .

در واقع واحد واکشی دستورالعمل باید تعداد زیادی از دستورالعمل ها را از cacheمیان بلاک های اصلی در هر سیکل ، واکشی کند . عموما این بلاک های اصلی جایگزین موقعیت های غیر پیوسته در cache می شوند . بیش از یک دسترسی به cache باید به منظور واکشی همه پلاک های اصلی مورد نیاز ، صورت گیرد . مکان های واکشی غیر پیوسته در تعداد دستورالعمل ها محدود است . از این رو cache را همکار مناسبی برای واکشی چندین بلاک اصلی در هر سیکل محسوب می گردد .

Trace cache (اثر کش) تکنیک ریز معماری برای افزایش پهنای باند واکشی دستورالعمل یک پردازنده superscalar می باشد . پردازنده ، جریان دستورالعمل اجرا شده را جمع آوری کرده و آنها را در خطوط Trace cache با آدرس های مسیر انشعاب ذخیره می نماید . در این روش ، دستورالعمل های واکشی شده، از مسیر اجرای غیر پیوسته در موقعیت های پیوسته در Trace cache جایگزین می گردد . هنگامی که یک خط Trace cache به مسیر اجرای بعدی ، ارجاع داده می شود ، خط کلی را می توان در طی یک دسترسی به cache واکشی نمود . [1] . مکانیسم های واکشی و اجرا بوسیله بافرها برای مثال صف ها ، مکانیسمهای رزرو و . . . مجزا شده اند . مکانیسم واکشی دستورالعمل مانند یک تولید کننده ای که عمل واکشی دیکد و غیره را انجام می دهد ، عمل می کند .[2] کارایی یک پردازنده تک thread به شدت وابسته به میانگین تعداد دستورالعمل های مفیدی است که می توان آنها را درسیکل واکشی نمود . trace cache ها دربرآورده سازی این هدف با ذخیره دستورالعمل ها بصورت پویا بسیار مؤثر هستند [3] . trace cache ها بعنوان راه حل موثری برای مشخص کردن محدودیت های مکانیسم واکشی پیشنهاد شده اند . این cache ها بلاک های اصلی دستورالعمل را به روشی که مجدداً در طی اجرای برنامه قابل بازیابی باشند ، ذخیره می کند . هر خط trace یک دنباله از بلاکها را بااطلاعات نزدیک به بلاکها ذخیره می نماید . برای پیش بینی انشعاب بعدی درطی هر واکشی چندین پیش بینی کننده انشعاب مورد استفاده قرار گرفته است . [4] دراین مقاله, ما به ترتیب بخش های زیر را دنبال می کنیم :

2-1-بخش اول

عموماً ، هدف اصلی c و برنامه های Fortran که دارای انشعابات غیر مستقیم کمتر ، انشعابات شرطی و بازگشت های تابعی کمتری می باشند ، این است که انشعابات قابل پیش بینی باشند . اگرچه ، فرکانس استفاده از انشعابات غیر مستقیم بیشتر از کاربردهای آینده است ، اگر چه انشعابات شرطی در بیشتر application ها ، ادامه می یابند . برای مثال برنامه های شیء گرا object oriented ، از انشعابات غیر مستقیم بیشتری برای پیاده سازی فراخوان های تابع مجازی استفاده می نمایند . کتابخانه های نیک شده بصورت دینامیکی (dll) با استفاده از انشعابات غیر مستقیم فراخوانی می شوند . نهایاتا افزایش تعداد زبانهای برنامه نویسی با استفاده از ماشین های مجازی (VMS) نظیر java به تکنیکی نیاز دارند که تعداد زیادی انشعاب غیر مستقیم را دارا باشد . بنابراین نرخ دقت در پردازنده های ILP افزایش می یابد .

ما مشاهده می کنیم که در یک پردازنده ILP با Trace cache، امکان افزایش دقت پیش بینی انشعاب غیر مستقیم برنامه ها با استفاده از Trace cache به جای استفاده از بافر مسیر انشعاب وجود دارد . بعلاوه ، استفاده از این طرح مستلزم ایجاد ساختار بسیار ساده سخت افزاری بدون استفاده از جداول اضافی می باشیم .

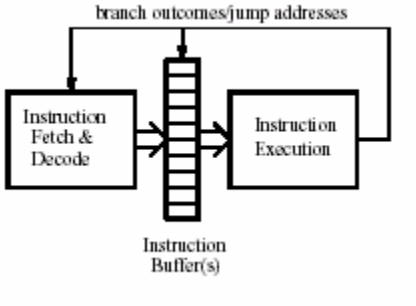
در این مقاله نشان می دهیم چگونه Trace cache می تواند برخی از محتوای اطلاعاتی مورد استفاده بوسیله پیش بینی کننده های غیر مستقیم دوسطحی را تقسیم کند . اگر چه افزایش دقت، کمتر از انشعاب غیر مستقیم دو سطحی می باشد و هزینه نیز کمتر است . اگر کسی قصد ساخت یک Trace cache را داشته باشد ، باید قادر باشد پیش بینی غیر مستقیم را با یک BTB نرخ کم با هزینه غیر اضافی انجام دهد . ما در این مقاله برای مقایسه پیش بینی کننده های انشعاب غیر مسقتیم دو سطحی یا nent trace تلاشی نکرده ایم . ما منحصراً مزایای دو سطحی را با استفاده از Trace cache نشان داده ایم . مبنی بر مشاهدات اولیه را هکارهایی را برای افزایش دقت ارائه می دهیم ۱) پیشنهاد می کنیم آدرس مسیر انشعاب غیر مستقیم را در Trace cache بهنگام سازیم . ما استفاده از سایت بهنگام سازی را پیشنهاد می نمائیم . ۲) نشان می دهیم بهنگام سازی با شمارنده ۲ بیتی در انتهای هر خط انجام گیرد.

2-2-بخش دوم

طبق شکل ۱ ، پردازنده های superscalar با کارایی بالا به مکانیسم واکشی دستورالعمل و مکانیسم اجرای دستورالعمل تقسیم می شود . موتور اجرای دستورالعمل مصرف کننده ای است که دستور العمل ها را از بافر حذف

کرده و سپس آنها را اجرا می کند . وابستگی های کنترلی (انشعاب ها و jump ها) یک مکانیسم باز خورد را میان تولید کننده و مصرف کننده بوجود می آورد .

پردازنده هایی که از این تکنینک ها استفاده می نمایند ، سعی در اجرای دستورالعمل بصورت موازی با هم دارند . بافر های بزرگ برای حفظ یک قالب دستور العملی برای اجرا بصوورت موازی مورد استفاده قرار می گیرند . برای فعال نمودن اجرای همروند دستورات ، موتور اجرایی از واحد های کاربردی زیادی که بصورت موازی با هم قرار گرفته اند عمل می نماید . موتور واکشی انشعاب ها را برای اجرای پیوسته دستورات به درون قالب window ارسال می کند . هدف از این طرح افزایش مقیاس این تکنیک هاست :



شکل ۱ :تجزیه موتورهای عمل گر واکشی

Dispatch وسیعتر ، قابهای بزرگتر ، ثابت های فیزیکی بیشتر، واحد های کاربردی بزرگتر و . . . . برای حصول این اهداف ، برقراری تعادل میان همه قسمتهای پردازنده بسیار مهم است .

در این مقاله ، ما پهنای باند واکشی دستورالعمل را برای افزایش کارایی مورد بررسی قرار داده ایم . کارایی واکشی دستورالعمل وابسته به تعدادی از فاکتورها می باشد . نرخ اصابت و دستورالعمل cacheو دقت درش بینی انشعاب مسائل بسیار مهمی است که در چنین واکشی دستورالعمل مورد استفاده قرار می گیرد . در این مقاله ، فاکتورهای دیگری را که در هر سیکل واکشی مؤثر است بررسی نموده ایم :

بازدهی انشعاب : اگر فقط در هر سیکل یک انشعاب شرطی اجرا شود ، می توان window را در نرخ فقط یک بلاک اصلی در هر سیکل افزایش داد . پیش بینی چندین انشعاب در هر سیکل باعث می شود بازدهی دستورالعمل کلی بالاتر گردد .

تنظیم دستورالعمل غیر پیوسته – بدلیل انشعابات و پرش ها jumpها ، و دستورالعمل ها باید در طی هر سیکل داده شده واکشی گردند . از این رو ، باید مسیر های مناسب و منطقی برای واکشی و تنظیم بلاک های اصلی غیر پیوسته و عبور آنها به گذر گاه وجود داشته باشد . فقط کافی نیست که دستورالعمل ها در cache حضور داشته باشند ، بلکه باید آنها بصورت موازی قابل دسترسی باشند .

دوام واحد واکشی ، دوام گذرگاه تاثیر مهمی روی کارایی پردازنده دارد . این تاثیر ناشی از هزینه پر کردن مجدد گذرگاه بعد ازکنترل غیر صحیح می باشد . درمورد واجد واکشی ، هزینه شروع واکشی مجدد بعد از کم کردن انشعاب ، Jump یا دستورالعمل cache بسیار مهم است . نیاز به بازدهی انشعاب بیشتر و تنظیم دستورالعمل غیر پیوسته دوام واحد واکشی افزایش خواهد یافت .

واحدهای واکش کنونی به یک پیش بینی انشعاب درهرسیکل محدود شده می تواند ۱ بلاک اصلی را در هر سیکل واکشی کند .

داده های موجود درجدول ۱ ، میانگین سایز بلاک های اصلی را که در حدود ۴یا ۵ دستورالعمل برای کدهای صحیح می باشد ، نشان می دهد . واکشی یک بلاک اصلی درهر سیکل برای پیاده سازی بیش از ۴ دستورالعمل درهر سیکل کافی می باشد . اگر ما چندین پیش بینی انشعاب را معرفی می نمائیم ، واحد واکشی در نهایت می تواند چندین بلاک اصلی پیوسته را درهر سیکل واکشی نماید . داده هایی که برای چندین دستورالعمل میان انشعابات گرفته شده مورد استفاده قرارمی گیرند تا حدی به دللی فرکانس انشعابات محدود شده است . بنابراین ، اگر یک انشعاب گرفته شود ، واکشی دستورات زیرمسیر قبلی درهمان سیکلی که انشعاب واکشی شده است ، مورد نیاز می باشد .

جدول (۱) : انشعاب و بلاک اصلی

Benchmark	taken %	avg basic block size	# instr between taken branches
eqntott	86.2%	4.20	4.87
espresso	63.8%	4.24	6.65
xlisp	64.7%	4.34	6.70
gcc	67.6%	4.65	6.88
sc	70.2%	4.71	6.71
compress	60.9%	5.39	8.85

2-3-بخش سوم

استفاده از trace cache ، اگر دارای سایز محدود شده ای نباشد ، بسیار مؤثر است . اگر چه trace cache های

کوچک به علت زمان دسترسی و کارایی حافظه بسیار کاربردی تر است : بلاک های اصلی را می توان در trace cache های مختلفی مشاهده نمود ، بعضی از trace ها حاوی ماکزیمم دستورالعمل ها نمی باشد و بلاک مورد در خواست ممکن است درون trace باقی مانده و غیر قابل دسترس گردد . بنابراین کلید برای استفاده مؤثر از trace cache های بسیار کوچک ، ارائه روشی برای تغییر ساخت آنهاست . این مقاله برروی دومین قسمت منطقی مورد نیاز برای ساخت trace ها متمرکز شده است . فیلترینگ هم اکنون روش پیشنهادی برای افزایش تاثیر سایز trace cache میباشد . دراین روش پیشنهاد می شود ، همه trace هایی که حاوی انشعابات هستند ذخیره شوند . پس باید آنها را مبنی بردرصد استفاده نشان فیلتر کرد . trace cache به دو بلاک اصلی تقسیم شده است : (FTC) ,(MTC) . همه trace ها روی FTC نوشته می شود . اما فقط trace های موجود در MTC مفید می باشند . موفقیت این روش فیلترینگ مبنی برمشاهده آنهاست ، بیشتر trace های ساخته شده و قرار گرفته در trace cache قبل از eviction مورد استفاده قرارگرفته و بیشتر دستورالعمل ها از مجموعه trace های کوچک اجرا می شود . پیشنهاد شده است که به منظور فیلتر کردن trace هایی که غالباً مورد استفاده قرارگرفته اند ، نمونه برداری شده و لوکالیتی آنها نشان داده شود . این مقاله نوع جدیدی از تکنیکها ی فیلترینگ را که مبنی برنمونه برداری آماری trace ها می باشد معرفی می نماید .این نوع فیلتر ها می خواهند کیفیت trace ها رادریک trace cache کوچک افزایش دهد . این مقاله کارایی و توان فیلتر نمونه برداری اصلی (sf) وورژن ارتقاء یافته آن را به همراه مقایسه آن با ftc –mtc تحلیل وبررسی می کند .

4-2-بخش چهارم

پردازنده های superscalar قادر است تعداد زیادی از دستورالعمل ها را درقالب یک سیکل واحد اجرا نماید . اگر چه مکانیسمهای واکشی دستورالعمل باید دستورالعمل ها را با مکانیزم موازی اجرا نماید . دقت پیش بینی انشعاب ، نرخ اصابت در cache و قواسیهای اجرای دینامیکی فاکتور های تاثیر گذار در کارایی مکانیسم واکشی می باشند .

آدرس بلاک شروع و بیت های آن برای تعیین حضور trace مورد نیاز در trace cache مورد استفاده قرار گرفته است .

اگر trace cache حضور داشته باشد ، آن از trace cache یک سیکل واحد بازیابی شده است . درحالیکه trace cache ها به حصول کارایی بهتر کمک میکنند اما دارای محدودیت هایی نیز می باشند . برخی از این محدودیتها عبارتند از :

- دیده شده است که با استفاده از trace cache های بزرگتر ، مزایای مربوط به کارایی کاهش می یابد . درواقع مصرف توان افزایش می یابد .
- trace هایی که کاملاً شامل بلاک های پیوسته باشند احتمالاً درخطوط آدرس دهی cache موجود می باشند . حضور چنین trace هایی منجر به ایجاد افزونگی و هدر رفتن فضای حافظه خواهد شد .
- با توجه به اینکه ۸۰٪ زمان درزمینه ۲۰ ٪ از کدها صرف می شود ، ممکن است LRU روش بهتری باشد .

راهکار ما برای حل این مسائل استفاده از نموداری است که برای طراحی جایگزینی trace ها در trace cache مورد استفاده قرار می گیرد . هربرنامه کاربردی به پارتیشن های کوچکتر تقسیم می شود و هریک ازاین زیرمجموعه ها نمودار می شوند لوکالیتی درجریان نمونه افزایش می یابد .

کاربرد اصلی این پروسه تعریف و تشخیص trace هایی است که غالباً مورد استفاده قرارگرفته است .این پروسه درواقع برای تقسیم trace ها به دو سطح بصورت زیربسیار مهم هستند هرزمانی که یک trace به دورن trace cache آورده می شود جزئیات آن با trace های موجود درجدول چک می گردد.

اگر اطلاعاتی یافت نشود ، trace به درون cache وارد شده و شمارنده ان را تغییر می یابد . بنابراین هروقت trace مورد اصابت قرار می گیرد ، شمارنده اننیز افزایش می یابد . هنگامی که درصد استفاده آن به استانه t1 می رسد ، جزئیات آن به یک مدخل در cache نوشته می شود . اگر بعد ازاین trace از cache بازیابی شود ، مدخل جدول به اخرین مقدار بهنگام سازی می گردد . اگر بعد ازاین زمان ، همان trace به cache وارد شود ، این جزئیات به شمارنده انتقال داده می شود.

هنگامی که مقدرا شمارنده به دومین آستانه t2 برسد ، شمارنده بهنگام سازی می گردد . توجه کیند که شمارنده استفاده از آنها دارای مقادیر بزرگتر از t2 می باشد ، قادر به حذف شدن نمی باشند . اگر جدول با چنین مداخلی پرشود ، ماشروع به درخواست برای هر مدخل می کنیم . هنگامی که شمارنده به ۱۵ برسد ، پروازنده متوقف شده و

اطلاعـات بـه دورن فایـل حافظـه نوشـته مـی شـود . درایـن روش کل برنامه مورد استفاده قرار گرفته است .
منطـق جـایگزینی مبنـی براهمیـت trace هـا بـه نـسبت یکـدیگر اسـت . trace هـایی کـه شـامل انـشعابات برگرفتـه باشـد . بعنـوان بهتـرین کاندیـداهابرای جـایگزینی درنظرگرفتـه می شوند .
سربارسیستم روی سـخت افـزار اعمـال مـی شـود .درایـن مقالـه مـافقط طـرح مربـوط بـه معمـاری والگـوریتم هـای مـورد استفاده برای منطق جایگزینی راارائه می دهیم .

3-طراحی وظایف

طـرح پیـشنهاد شـده بـه انجـام تغییـرات مشخـصی درزمینـه طراحی موجود نیازمند است .

3-1-جدول نمونه

بـه منظـور جـدول بنـدی trace هـا مابـه یـک جـدول نمونـه نیـازداریم کـه اطلاعـات مربـوط بـه trace درآنهـا حفـظ و نگهـداری گـردد . ایـن جـدول حـاوی مـداخل بـسیرازیادی مـی باشـد کـه حـاوی trace cache درزمـان داده شـده مـی باشـد . هر مدخل باید فیلدهای زیر را دارا باشد :

4-آدرس های بلاک شروع

- بیتهـای پیـشنهاد بینـی مربـوط بـه بلاکهـای موجـود در trace ها
- فـاکتور اولویـت مربـوط بـه درصـد اسـتفاده trace در هریـک اززیـر مجموعـه هـای برنامـه .بـرای اجتنـاب از - wrap around این فیلد باید دقت کافی صورت گیرد .

بیت mask بـرای اینکـه بتـوان یـک مـدخل رابـرای جـایگزینی انتخاب نمود .

4-1-تغییرات trace cache

خـط traceباید بـرای طـرح پیـشنهادی ، دچـار تغییـرات و اصـلاحاتی گـردد . هرمـدخل در trace cache باید شـامل یـک فیلـد باشـد تـا درصـد اسـتفاده حفـظ گـردد . این فیلد مرتبط با فـاکتور اهمیـت اولـین زمـانی اسـت کـه یـک trace جـدول بنـدی شـده بـه درون trace cache آورده شده است .
یـک بیـت تـصمیم میگیـرد کـه trace مهـم اسـت یـا اینکـه دارای اهمیـت نمـی باشـد درصـد اسـتفاده ازایـن بیـت بـه تفضیل بیان شده است .

4-2- طرح جدول

ایـن طـرح شناسـایی مهمتـرین trace هـا دریـک از پارتیـشن های برنامه را انجام مـی دهـد . هنگـامی کـه نمونـه دراجراهـای متوالی مورد استفاده قرارگیرد ، ایـن نمونـه بایـد قـادر باشـد بـه مـا کمـک کنـد تـا traceهـای مهـم و غیـر مهـم را از هـم تـشخیص دهـد . بازیـابی trace هـای مـورد اسـتفاده کمـک مـی کنـد تـا سـر بارسیـستم درهنگـام سـاخت مجـدد trace کاهش یابـد ، همچنـین درصـد هـرز رفـتن فـضا درcache نیـز کاهش می یابد . این نتیجـه ازایـن حقیقـت مـشتق شـده اسـت کـه شـانس هـای زیـادی موجودنـد کـه trace هـایی را مـورد اسـتفاده قـرار میدهنـد کـه ممکـن اسـت جـایگزین trace هـای مـورد اسـتفاده متـداول موجـود در cache گردنـد . محققـان معتقدنـد کـه راهکـار پیـشنهاد شـده تنهـا hit هـای trace را افزایش نمی دهـد ، انهـا مـی تواننـد منطـق بـافرپر را ذخیره کنند .

4-3-ساخت جدول

جـدول بایـد جزئیـات را دربـاره traceهـای متـداول وبـه روش فشرده دارا باشد .
Trace کامل شده
هنگـامی کـه یـک trace بـرای اولـین بـار بـه درون trace cache آورده مـی شـود ، شـمارنده درصـد اسـتفاده آن بـا ۱ مقـدار دهـی اولیـه مـی گـردد . بـا هراصـابت ایـن شـمارنده افزایش می یابـد ووقتـی مقـدار آن بـه بـالا تـر از مقـدار اسـتانه t1 افـزایش یافـت ، یـک مـدخل بـرای ایـن trace سـاخته شـده و همـه جزئیـات نظیـر آدرس بـلاک شـروع ، بیـت هـای نمونـه انـشعاب و شـمارنده درصـد اسـتفاده بـه درون آن نوشـته می شود . بعد از اینجـا اصـابتها شـمارنده هـای درصـد اسـتفاده را در trace cache و جـدول بهنگـام سـازی مـی کنـد . حتـی اگـر ایـن trace از cache حـذف شـود ، مـدخل جـدول بـدون اسـیب بـاقی مـی مانـد . در زمـان بعـدی همـان traceآورده مـی شـود شـمارنده مربـوط بـه درصـد اسـتافده آن با مقداری جـدول ، مقـدار دهـی اولیـه م یگـردد . وبـا هربـار اصـابت مقـدار ان افـزایش مـی یابـد . هنگـامی کـه ایـن درصـد اسـتافده بـه مقـدار اسـتانه t2 برسـد ، تـصمیم میگیـریم کـه matvre trace شـود و بیـت mask مـدخل آن در جـدول بـا تنظـیم شـود . توجـه داشـته باشـید کـه همـه مـداخل بایـد update گردد . استراتژی وقفـه کـه مـا مـورد بحـث قـراردادیم مـارا درسـاختن جـدول بـرای هـر پارتیـشن برنامـه یـاری مـی نماید .

4-4-جایگزینی مدخل جدول

طبیعی است که همه مدخل جدول نمونه پرشوند و روشن مناسبی برای جدول بندی دیگر traceهائی که ممکن است درصد اصابت به آنها از t1 بیشتر باشد درنظر گرفته شود . ماشاخصی رابرای مداخل جدول ساخته و جایگزینی را انتخاب می نمائیم .

4-5-استراتژی وقفه

همه مداخل مربوط به جدول نمونه تکمیل خواهد شد و هیچ مدخلی برای جایگزینی موجود نم یبادش از این رو همه درخواست ها برای مداخل جدید محدود می شود . اگر این امر اتفاق بیفتد ما زا شمارنده ای استفاده می کنیم که عدد تکذیب ها را حفظ کند .

و هنگامی که مقدار آن به ۱۵ برسد ، زمان آن به درون فایلی در حافظه نوشته می شود . وقفه تولید شده باید بوسیله پردازنده سرویس دهی شود . در هنگام هریک از نوشتن های نمونه همه مداخل trace cache باید به صفر برسد .

4-6-درصد استفاده نمونه

نمونه را می توان در زمان اجرا مورد استفاده قرارداد . البته اگر شرایط اجرای برنامه یکسان باشد .

هنگامی که محیط اجرا تغییر نکند ، جدول مربوط به هر زیرمجموعه از برنامه به درون جدول نمونه بار گذاری می شود . هدف ازاین مبحث این است که ما مجاز به تعریف و تشخیص trace های مهم و غیر مهم باشیم . قبل ازاینکه یک خط بوسیله با فرساخته شود . جدول نمونه برای تعیین یک trace مهم در مورد بررسی قرارمی گیرد . اگر عمل انجام گیرد trace ساخته شده و سپس به درون cache بارگذاری می شود ، شمارنده درصد استفاده با فاکتور اهمیت تنظیم می شود و بیت اهمیت در trace تنظیم می شود . اگر عمل محدود صورت نگیرد یک مقدار تصادفی میان t1,o محاسبه می شود . این مقدار باید در شمارنده درصد استفاده trace ذخیره گردد ، این عمل با تنظیم مجدد بیت اهمیت درصورتی که traceمدخل cache را پیدا کند ، انجام می گیرد . هر زمانی که trace درcache مورد اصابت قرارگیرد ، فاکتور اهمیتش به ۱ کاهش می یابد . منطق جایگزینی بعداً بحث قرارمیگیرد . هنگامی که همه خطوط trace cache بابیت اهمیتش تنظیم گردد . طرح زیرمورد استفاده قرارمیگیرد . اگر خط ساخته شده با منطق بافرپر یک trace مهم باشد . trace بافاکتور نهایی اش فعال میگردد . مدخل مرتبط بااین trace درجدول برای منعکس کردن آخرین مقدار فاکتور اهمیت تغییر می یابد .

اگر یک trace غیر مهم با منطق بافر پر ساخته شود . یک مقدار تصادفی میان محاسبه میگردد . اگرمقدار تصادفی trace جدید بزرگتر از فاکتور اهمیت نهایی باشد ، دومی با trace ساخته شده جدید جایگزین می گردد ، در غیر اینصورت traceجدید اصلاً ساخته نمی شود . در مورد trace جدید یک مدخل جستجو م یگردد و پس فاکتور اهمیتش بایک مقدار تصادفی محاسبه شده مقدار دهی اولیه می گردد . اگر در cache یک trace مهم یا بیش ازیک trace مهم موجود باشد ،یک ایندکس به trace cache جدید وارد می شود و traceها به روش گردش نوبتی مورد جستجو قرارمیگیرد .

در موارد بالا مقایسه بیتهای پیشگویی میان trace هایی که دارای فاکتور اهمیت اولیه هستند ، با شکست مواجه می شود .

trace هایی که شامل انشعابات برگرفته هستند ، مقدم ترند . اگر بیت های پیش بینی trace ها مشخص باشد ، یکی ازآنها بطور تصادفی انتخاب می شود .

5-مزایا

درهنگام استفاده از منطق جایگزینی مزایای زیادی حاصل می شود . مزایای حاصله مربوط به افزایش کارایی trace cache ها می باشد . می توان دید که کارایی trace cache با افزایش سایز افزایش مییابد .

نتایج مورد استفاده دراین بخش از طراحی trace cache برگرفته شده است . ipc نشان داده شده دراینجا میانگین برای یک مجموعه با برنامه نسبت م یباشد . افزایش سایز کش باعث می شود تعداد traceها درهر نقطه اززمان درکش افزایش یابد . این افزایش یابد . این افزایش باعث زیاد شدن اصابتها و افزایش کارایی می گردد . اگر چه آشکار است که افزایش سایز اقتصادی نم یباشد . زیرا مصرف نیروی کش زیاد می گردد .

این طرح سعی دارد هنگامی که کش کاملاً پراز trace های متداول است ، انها را مورد استفاده قراردهد . واکشی بعدی برای trace متداول باعث ساخت مجدد trace میگردد .

trace مورد استفاده باید خیلی زود از cache حذف گردد . مشاهده شده است که چنین استفاده افراطی از trace cache نباید درطرح ما مجاز باشد .

هنگامی که دو trace برای جایگزینی درکش با هم رقابت می کند . طرح ما اولویت رابه traceهای می دهد که

زودتر ظاهر شده اند . دلیل این امر مبنی بر دو مشخصه زیراست .

i پیش بینی کننده های چندین انشعاب به دلیل اینکه مجبورند تعداد انشعابات را در طرح پیشگویی کنند ، دارای دقت کمتری می باشند .

6-نتیجه

در این مقاله ، تاثیرات استفاده از Trace cache را روی پیش بینی انشعاب غیر مستقیم در پردازنده های ILP مورد بررسی قرار داده ایم . اگر آدرس های مسیر در خطوط Trace cache برای پیش بینی انشعاب استفاده گردد ، دقت را می توان متفاوت از بافر آدرس ساده بررسی نمود . دلیل اصلی این است که Trace cache حاوی جریان کنترلی اخیر برنامه می باشد .

ما یک مکانیسم ساده بهنگام سازی آدرس های انشعاب غیر مستقیم در خطوط Trace cache ، ما دقت را می توانیم با صرف کمترین هزینه یا حتی بدون صرف هزینه افزایش دهیم . ما پیکربندی های متفاوتی را به همراه استراتژی های مختلف نظیر فشرده سازی ، اضافه کردن شمارنده ۲ بیتی ، مشارکت مجموعه cache ، سایز cache سایز خط cache و تاثیرات مثبت بر منفی هر پیکربندی بر استراتژی را مورد بررسی قرار داده ایم .

نتایج آزمایش ما متوسط دقت ها رمونیک را در میان چندین انشعاب با استفاده از یک مدل Trace cache به همراه پارامترهای Trace cache نشان داده شده است که می توان آن را با استفاده از بهنگام سازی نسبت به BTB ، ۳۵/۷۵٪ و نسبت به حالت غیر بهنگام سازی ۱۱/۷۷ ٪ افزایش داد .

اگر چه این مقاله تاثیر Trace cache را روی پیش بینی غیر مستقیم که ممکن است تاثیر مشابهی روی دیگر فرمها داشته باشد ، مورد بررسی قرار داده است . برای مثال ، مقدار بار ممکن است در صورت وجود مداخل مجزا برای هر یکی تحت تاثیر قرار گیرد . محاسبه این تاثیر برای طراح پردازنده که انتظار آدرس دهی با استفاده از Trace cache را دارد بسیار مهم است . [1]

مراجع

[1]- Davin Greeg ، ژورنال، برچسب دستورات برابر ، دانشگاه ایرلند ، سال ۲۰۰۶

[2]- james smith ، steve Bennett ، eric rotenberg ، علوم فرورفتگی کامپیوتری ، دانشگاه دیسکونسین ، سال ۲۰۰۳ .

[3]- micaeel behar, Avi Mendelson , Avinomam ، دانشگاه مهندسی الکترونیک هیفا اسرائیل ، سال ۲۰۰۵ .

[4]- Saisuresh Krishnakmuaran ، دانشگاه مهندسی الکترونیک و کامپیوتر، کنیا هند ، سال ۲۰۰۴