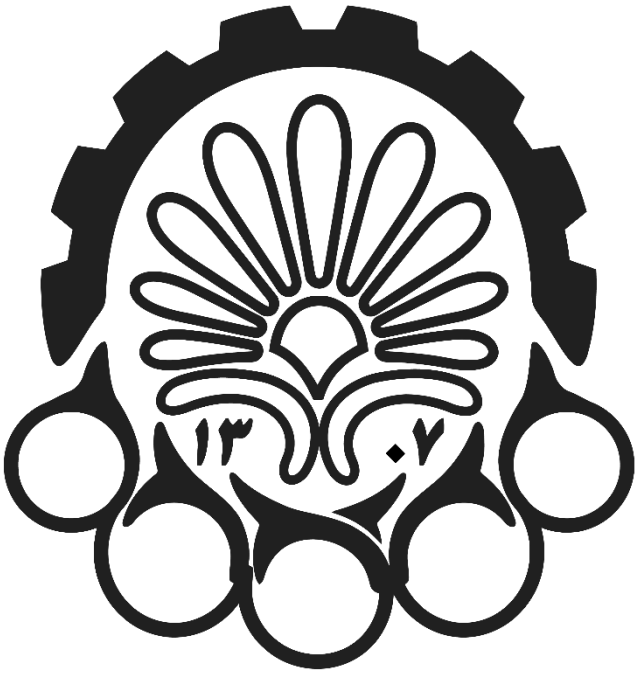
معماری کامپیوتر پیشرفته

دکتر حامد فربه



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

امیررضا حسینی

۴۰۲۱۳۱۰۶۰

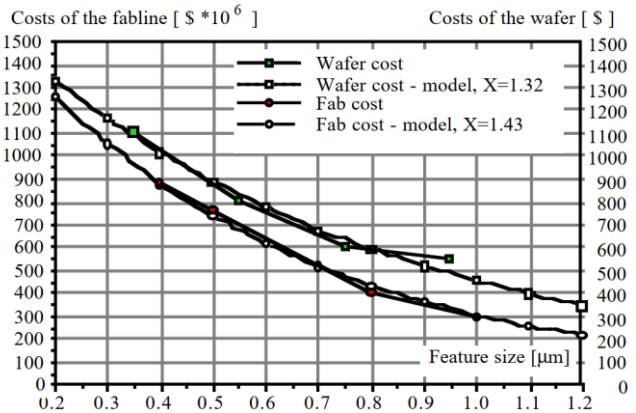
تمرین شبیه‌سازی اول

چکیده— تکامل علوم کامپیوتر منجر به توسعه تعداد زیادی شبیه‌ساز معماری کامپیوتر شده است، هر کدام با ویژگی‌ها و قابلیت‌های منحصر به فرد خود. این گزارش در نظر دارد تا مقایسه‌ای عمیق از سه شبیه‌ساز از این دست، یعنی GEM5 ، MARSS و SimpleScalar و غیره ارائه دهد. این مقایسه بر اساس چندین پارامتر کلیدی از جمله عملکرد کلی، مزایا، سیستم عامل میزبان و فرآیند اجرای هر شبیه ساز و دریافت نتایج ورودی خروجی و محیط گرافیکی خواهد بود. هدف ارائه درک جامع از این ابزارها است، تفاوت ها و شباهت های آنها را برجسته می کند. گزارش با مروری بر هر شبیه ساز آغاز خواهد شد، پس از آن تجزیه و تحلیل دقیق عملکرد آنها را دنبال خواهد کرد. سپس به بررسی مزایای هر شبیه ساز خواهد پرداخت، بینش هایی را در مورد آنچه هر یک را منحصر به فرد می کند، فراهم می آورد. سیستم عامل میزبان برای هر شبیه ساز نیز مورد بحث قرار خواهد گرفت، همچنین دستورالعمل های اجرای هر شبیه ساز و تفسیر نتایج. علاوه بر این جنبه ها، گزارش به بررسی سایر تفاوت هایی خواهد پرداخت که این شبیه ساز ها را از یکدیگر جدا می کند. هدف ارائه نگاه جامع به این ابزارها است، کمک به کاربران برای تصمیم گیری آگاهانه در مورد شبیه ساز مناسب ترین نیاز آنها است. گزارش با خلاصه ای از یافته ها و توصیه های برای تحقیقات آینده در این زمینه به پایان خواهد رسید.

کلمات کلیدی — معماری کامپیوتر، شبیه‌سازی، *gem5* ، *X86*

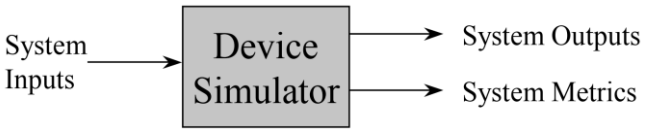
أ. مقدمه

توسعه و آزمایش سیستم های واقعی می تواند هزینه بسیار بالایی داشته باشد، که باعث می شود معماران کامپیوتر برای ارزیابی گزینه های طراحی مختلف، به تکنیک های شبیه سازی و مدل سازی تکیه کنند. تخمین زده شده است که حدود ۹۰٪ از مقالات ارائه شده در کنفرانس های برتر، شبیه سازی را به عنوان روش ارزیابی عملکرد استفاده می کنند. با این حال، بسیاری از نظرسنجی های موجود در مورد شبیه سازی های معماری کامپیوتر منسوخ شده اند و شامل آخرین شبیه سازها نمی شوند. بنابراین، شبیه سازها نقش بزرگی در پیشرفت جهان امروز ایفا می کنند. ساخت یک برد مدار چاپی یکپارچه، که هزینه آن در بسیاری از موارد برای مقیاس بزرگ تقریباً ۱ تا ۲ میلیارد دلار است[۱]، بدون اطمینان قبلی از کاربردهای دقیق آن، چالش قابل توجه ارائه میدهد. و همچنین اگر ادعایی در رابطه با پیشرفت گوشه‌ای از علم کامپیوتر به صورت تئوری وجود داشته باشد، ابتدا آن را با شبیه‌سازها در چند حالت مختلف با جزئیات کمتر تست می‌کنیم و سپس به دنیای واقعی و روی تراشه پیاده سازی میکنیم. در نتیجه، استفاده از شبیه ساز ها و شبیه ساز های نرم افزاری در این زمینه ضروری است. شکل ۱ نموداری را در رابطه با این اطلاعات نشان می دهد.



شکل ۱ هزینه ساخت یک ویفر[1] بر اساس سال[3].

عدم وجود اعتبار سنجی عملکرد شبیه ساز ها ممکن است منجر به خطاهای آزمایشگاهی شود که منجر به نتیجگیری های نادرست می شود. این کار یک دقت شبیه ساز و مقایسه عملکرد بعضی از شبیه ساز های معمار چندگانهٔ x86 رایانهٔ جدید را فراهم می‌آورد، مثل gem5[2] ، MARSS[3]، Dinero[4] در جنبهٔ کلّی مقایسهٔ عمومی عملکرد، نقطهٔ قوّت و ضعف و بازبینی فنی سامانهٔ عامل پذیرش، ورودی / خروجی، تجربهٔ کاربر پایانی. شکل ۲ یک طرح کلی سادهٔ یک شبیه ساز معماری کامپیوتر را نشان می‌دهد.



شکل ۲ معماری کلی شبیه‌ساز های کامپیوتر.

ب. مقایسه جامع

i. عملکرد کلی

عملکرد یک سیستم بیشتر به سطح برنامه و نیازمندی های سیستم بستگی دارد. این موضوع به صراحت در برگه داده خاص شبیه ساز نشان داده شده است. انواع مختلفی از شبیه سازها وجود دارند، هر کدام با قابلیت های منحصر به فرد خود. به عنوان مثال، برخی از شبیه سازها، مانندgem5 ، شبیه سازهای سیستم کامل هستند که می توانند کل سیستم را شبیه سازی کنند. از طرف دیگر، شبیه سازهایی مانند Deniro وجود دارند که فقط بخش های خاصی از یک سیستم را شبیه سازی می کنند. انتخاب شبیه ساز می تواند تأثیر قابل توجهی بر طراحی کلی سیستم داشته باشد. بنابراین، حائز اهمیت است که یک شبیه ساز را انتخاب کنید که با نیازمندی های خاص سیستم مورد نظر ما همخوان باشد. در مرجع [۶]، نویسندگان یک سیستم مشابه Core i7 را شبیه سازی می کنند، که بر اساس معماری سامانهٔ X86 است. آنهٔ مقایسهٔ مختلف شبیه ساز ها و نتایج آن ها را ارائهٔ می دهند. مهٔ توجهٔ این نتایج با پیکربندی های مختلف قابل تغییر قابل توجهٔ است. بنابراین، در حالی کهٔ این نتایج بینش های ارزشمندی فراهم می کنند، باید با احتیاط تفسیر شوند زیرا تغییر پیکربندی ها می تواند منجر به نتایج کاملاً متفاوت شود. در [۷]، عملکرد Gem5 ارزیابی و با اجرای سخت افزار واقعی مقایسهٔ می شود. بستهٔ به نوع نرم افزار، عدم تطابق بین شبیه سازی Gem5 و اجرای سخت افزار واقعی در بازهٔ ۰.۴۷٪ تا ۱۷.۹۴٪ است. این به خاطر آن است کهٔ Gem5 رفتار زمانی حافظهٔ DDR را به درستی مدل نمی کند. در [۸]، عملکرد بین Gem5 و Qemu مقایسهٔ می شود. نتیجهٔ کهٔ کدام بهتر است، بستگی به نشانگر انتخاب شده دارد. ولی در کُلّ، CPU ARMخارج از ترتیب Gem5 زمان اجراء را به مقادیر واقعی زندگی نزدیک می کند. شبیه سازی بسیار دقیق Gem5 یکی از دلایل انتخاب ما است. در نتیجه، هنگام طراحی یک سیستم، ضروری است که سطح برنامه، نیازمندی های سیستم و انتخاب شبیه ساز را در نظر بگیرید. درک کامل از این عوامل می تواند در دستیابی به عملکرد سیستم بهینه کمک کند. باید توجه داشت که قدرتمندترین شبیه‌سازها در این زمینه را یعنی gem5 و deniro توسط آقای Mark D. Hill و تیمش طراحی و توسعه یافته است.

ii. مزایا و معایب

اصولا در دیتاشیت‌های اولیه در این باره زیاد صحبت نمیشود و در واقع باید در مقاله‌های مقایسه کنند دنبال این مطالب بگردیم. مثلا همانطور که میدانیم از مزایای شبیه‌ساز SimpleScalar سادگی کار با آن است و از معایب آن قدیمی بودن این شبیه‌ساز است. همچنین بعضی از شبیه‌سازها مخصوص اجرای برنامه نیستند و CPU را مدل نمیکنند و فقط مخصوص حالت خاصی از قسمت‌های دیگر مانند انواع مختلف حافظه هستند.

[1]wafer

شکل ۳ مقایسه‌ای جامع بین شبیه‌ساز های مطرح معماری X86

Simulator	Supported hosts (ISA/OS)	Category	Supported Processor Models	MultiCore Support
Gem5	x86, ARM, SPARC, Alpha, PPC/Linux,MacOSx,Solaris,OpenBSD	FS/MOD/TIM (cycle-accuate)	IO, OOO	yes (HMP)
Multi-2sim	x86/Linux	UM/MOD/TIM	OOO, multithreaded	yes (HMP)
Sniper	x86/Linux	parallel UM/TIM (interval simulation)	IO, OOO, SMT cores	yes (HMP)
PTLsim	x86/Linux	FS/TIM simulator (cycle-accurate)	OOO	yes
ZSim	x86/Linux	parallel UM/TIM	IO, OOO	yes (HMP)
Simple-Scalar	x86/Linux, Win2000, SPARC/Solaris	UM/ED/TIM	OOO	no
SIMFLEX	x86	FS/MOD/TIM (decoupled functional and timing)	OOO	yes
SIMICS	Alpha, PPC, UltraSparc, x86/Linux, Windows	FS/functional	functional	yes
Augmint	x86/Unix and Windows NT	ED/TD	functional	yes
Mc-SimA+	x86/Linux	UM/TIM (decoupled functional and timing)	IO, OOO	yes (HMP)
GEMS	x86/Linux, AMD64-linux, and SparcV9 (Solaris 8)	FS/TIM (decoupled functional and timing)	OOO	yes
MARSSx86	x86-64/Linux	FS/TIM	IO, OOO	yes
Graphite	x86/Linux	parallel UM/TIM (decoupled)	IO	yes
OVPsim	x86/Windows and Linux	functional	IO	yes
Flexus	x86/Linux	FS/TIM/ED (cycle-accurate)	IO, OOO	yes
Zesto	x86	UM/TIM cycle-level	OOO	yes
McPAT	x86/Linux	power, area and TIM	IO, OOO, NOCs	yes (HMP)
Note: UM=user mode, FS=full-system, ED=execution-driven, EvD=event-driven, TD=trace-driven, TIM=timing, MOD=modular, IO=in-order, OOO=out-of-order, HMP=hetrogeneous multiprocessor				

به عنوان مثال برای شبیه‌ساز SimpleScaler مزایایی وجود دارد که آن را از رقیبان خود متمایز میکند. مزایای اصلی شامل قابلیت گسترش (با شامل شدن منبع برای همه چیز: کامپایلر، کتابخانه ها، شبیه سازها)، قابل حمل (در میزبان، هدف مجازی بر روی اکثر جعبه های مشابه Unix اجرا می شود. در هدف، شبیه سازها می توانند چندین ISA را پشتیبانی کنند)، جزئیات زیاد (شبیه سازهای محرک اجرا، پشتیبانی از اجرای مسیر اشتباه، کنترل و حدس و گمان داده، و غیره...) و عملکرد بالا (بر روی P4-1.7GHz: Sim-Fast: 10+ MIPS و Sim-OutOrder: 350+ KIPS) است.

iii. شبیه‌سازهای *Cycle-accurate*

یک شبیه ساز دقیق چرخه، نوعی از برنامه کامپیوتری است که رفتار یک سیستم کامپیوتری را بر اساس چرخه به چرخه شبیه سازی می کند. مثلا با کامپیوتری که با ۲.۴ گیگاهرتز کار میکند، باید کلاک به کلاک سیستم را شبیه‌سازی کند که زمان به سزایی میشود. مثلا دو سه دقیقه از اجرای برنامه در کامپیوتر شبیه‌سازی شده ممکن است در عمل حدود دو الی سه ماه زمان بخواهد. این به این معنی است که عملکرد سیستم را تا به هر چرخه ساعت تقلید می کند. به همین دلیل در مقالات مطرح شده در حوزه معماری کامپیوتر، مثلا چیزی در حدود یک میلیارد دستور را روی شبیه‌ساز تست میکنند که تقریبا معادل با نیم ثانیه از فعالیت عادی سیستم هست تا مطمئن شوند که درست کار میکند و معتبر است. این اغلب در طراحی و توسعه سخت افزار جدید کامپیوتر استفاده می شود، اجازه می دهد مهندسان عملکرد یک سیستم را قبل از ساخت فیزیکی آن پیش بینی کنند. Gem5 ، SimpleScalarو MARSS نمونه های این دسته از شبیه ساز ها هستند.

ت. مقایسه فنی

همانطور که قبلاً اشاره شد، چندین عامل وجود دارد که می توان با اندازه گیری آنها، تفاوت های دقیق در هر کاربرد شبیه سازها را درک کرد. بنچمارک ها به عنوان ابزاری عمل می کنند که این درک را تسهیل می کنند. به عنوان مثال، هنگام بررسی بنچمارک های تعیین شده، خطای مطلق میانگین درصدی[1] در مقادیر دستورات در هر چرخه [2] (با استثنای نقاط پرت) نسبت به اجراهای سخت افزار واقعی به شرح زیر میباشد:

- Sniper: 24.8%
- gem5: 35.4%
- MARSSx86: 32.0%
- ZSim: 27.5%

برای بنچمارک‌هایی با مقادیر صحیح مقدار MAPE به شرح زیر میباشد:

- Sniper: 17.6%
- gem5: 37.1%
- MARSSx86: 22.16%
- ZSim: 22.59%

برای بنچمارک‌هایی با مقادیر اعشاری نیز این مقدار به صورت زیر تعریف میشود:

- Sniper: 24.8%
- gem5: 35.4%
- MARSSx86: 32.0%
- ZSim: 27.5%

این بنچمارک ها پارامترهای یک سیستم شبیه سازی شده را در طول زمان تغییر می دهند و آن را تحت شرایط مختلف آزمایش می کنند. این امکان را فراهم می کند تا تحلیل جامعی از عملکرد سیستم تحت سناریوهای مختلف ارائه دهد، بنابراین اطلاعات ارزشمندی در مورد عملکرد و کارآیی آن فراهم می کند. بنچمارک ها نقش حیاتی در درک عملکرد شبیه سازهای مختلف و کاربردهای آنها دارند. آنها یک سنجش کمّی از عملکرد سیستم فراهم می کنند، به ما اجازه می دهد تصمیمات آگاهانه ای در مورد انتخاب شبیه ساز برای نیازهای خاص سیستم بگیریم.

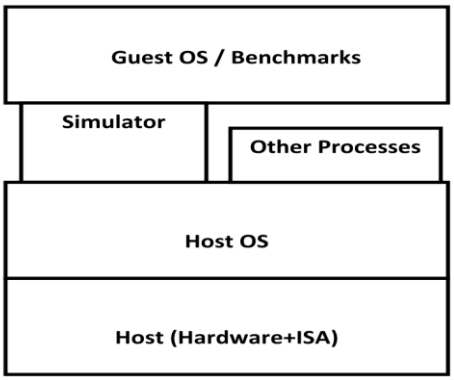
معمولا تنها توان و یا عملکرد تنها به طور کلی برای اکتفا به بهبود عملکرد و مقایسه چند سیستم کافی نمیباشد، لذا اکثر شبیه‌سازها پارامترهای پیچیده‌تری مانند EDP و PDP که شبیه‌سازها این پارامترها را هم گزارش میکنند.

i. سیستم‌عامل میزبان

در دنیای پیشرفته ی فناوری امروز، تقاضا برای سیستم عامل های متنوع بسیار زیاد است. بنابراین، سازگاری پلتفرم یک کیت ابزار شبیه ساز به یک عامل حیاتی تبدیل شده است که باید در نظر گرفت. این اطلاعات معمولاً در برگه داده های یک شبیه ساز یافت می شود. برای مثال، شکل ۳ فهرستی از سیستم عامل های پشتیبانی شده را نشان می دهد که می توانند شبیه ساز مشخص شده را میزبانی کنند. مهم است توجه داشت که در حالی که برخی از این سیستم عامل ها ممکن است نیاز به مجوز داشته باشند، دیگران به صورت رایگان در دسترس هستند. انتخاب سیستم عامل می تواند به طور قابل توجهی بر عملکرد و قابلیت های شبیه ساز تأثیر بگذارد. بنابراین، حیاتی است که سیستم عامل را انتخاب کنید که با نیازهای خاص شبیه ساز و سیستم طراحی شده منطبق باشد. این امر باعث بهینه سازی

[1] Mean Absolute Percentage Error
[2] Instructions Per Cycle

عملکرد و کارآمدی سامانه شبیه ساز شده خواهد شد. در نتیجه، درک و در نظر گرفتن سازگاری پلتفرم یک کیت ابزار شبیه ساز، گام حیاتی در طراحی و شبیه سازی سیستم است. این نه تنها بر عملکرد شبیه ساز تأثیر می گذارد، بلکه بر کارآمدی و اثربخشی کل سامانه طراحی شده نیز تأثیر می گذارد.



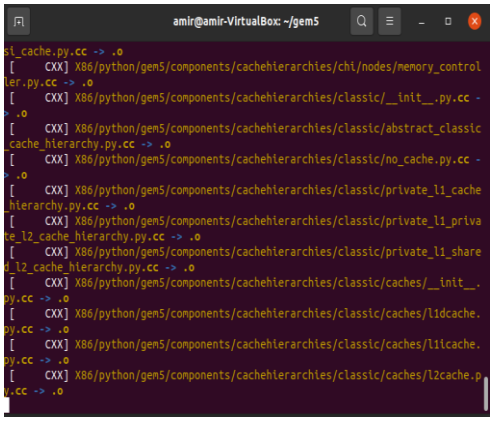
شکل ۴ دیاگرام اجرای شبیه‌ساز روی سیستم میزبان

۱. **شبیه‌ساز gem5**: Gem5 یک شبیه ساز سیستم کامل است که از بسیاری از ISA[1]ها با مدل های مختلف CPU پشتیبانی می کند. Gem5 از M5 برای مدل سازی دقیق CPU و از GEMS برای مدل سازی دقیق سیستم حافظه استفاده می کند. Gem5 در اصل از چهار مدل CPU پشتیبانی می کند: "AtomicSimple'، 'TimingSimple'، 'InOrder" و'O3'. AtomicSimple و TimingSimple معماری های تک چرخه غیر لوله ای هستند. 'InOrder' و 'O3' مدل های هسته IO و OOO لوله ای هستند. هر دو 'execute-in-execute' هستند، به این معنی که دستورالعمل ها فقط در مرحله اجرا پس از حل شدن تمام وابستگی ها اجرا می شوند. این ها را می توان برای شبیه سازی تعداد مختلف مراحل لوله، عرض صادرات و تعداد نخ های سخت افزار تنظیم کرد.

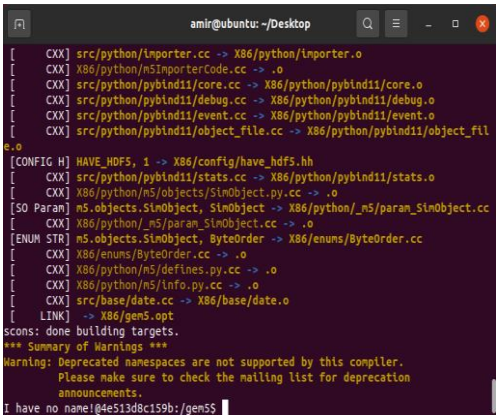
Gutierrez et al و Butko et al معادل بودن دقت gem5 را با مدل سازی سیستم های واقعی براساس ARM تأیید کرده اند[9]. پس از اعمال برخی تغییرات در شبیه ساز، به جز پیکربندی آن برای تطابق با برد آزمایشگاه (ARM Cortex A15)، Gutierrez et al توانستند خطای درصد زمان اجرای متوسط ۵٪ و خطای درصد زمان اجرای مطلق متوسط ۱۳٪ را برای نشانگرهای SPEC CPU2006 به دست آورند. Butko et al و همکارانش دقت gem5 را برای شبیه سازی یک سیستم چند هسته ای جاساز شده (ARM Cortex A9) تجزیه و تحلیل کرده اند. نشانگرهای مختلف مرتبط با بار کار علمی (SPLASH-2)، برنامه های رسانه ای (ALPBench) و پهنای باند حافظه (STREAM) برای تأیید استفاده شده اند. نتایج نشان می دهند که دقت از ۱.۳۹٪ تا ۱۷.۹۴٪ متغیر است.

۲. **نصب و استفاده از gem5:** با توجه به داکیومنت‌های موجود در سایت [10] میتوان این شبیه‌ساز را دانلود و نصب نمود و دستورات اولیه را با آن امتحان کرد. در ادامه، این شبیه‌ساز بر روی سیستم‌عامل اوبونتو ۲۰ تست و اجرا شده است.

بعد از نصب پکیج‌های مورد نظر (در اینجا پایتون و pre-commit) نوبت به بیلد کردن برنامه در پوشه مد نظر با استفاده از دستور `scons build/X86/gem5.opt -j <NUMBER OF CPUs ON YOUR PLATFORM>` است. (در این شبیه‌ساز، تعداد هسته‌های پردازنده اختصاص داده شده ۲ در نظر گرفته شده می‌باشد.) باید توجه داشت که ممکن است بیلد کردن آن روی یک سیستم معمولی، زمانبر باشد. شکل ۵ نمونه‌ای از فرآیند بیلد کردن را روی سیستم با مشخصات مذکور نشان می‌دهد. باید توجه داشت که میتوان از فایل داکر[2] هم برای سهولت بیشتر استفاده کرد.



شکل ۵ فرآیند بیلد کردن Gem5.



شکل ۶ نمونه از بیلد شدن موفق شبیه‌ساز gem5 توسط محیط داکر.

ii. زمان اجرا

هر شبیه ساز در محیط منحصر به فرد خود عمل می کند. برای مثال، شبیه ساز gem5 را در نظر بگیرید. پس از تغییر پیکربندی های سیستم و افزودن سیاست و ساختار کش مورد نظر، باید وابستگی های پایتون لازم را نصب کنید. پس از نصب این وابستگی ها، می توانید به ساخت و اجرای پروژه بپردازید. پس از زمان شبیه سازی خاص، خروجی آماری در کنسول نمایش داده می شود.

این خروجی اطلاعات ارزشمندی در مورد عملکرد و قابلیت های سامانه شبیه ساز شده فراهم می کند. شکل ۴ خروجی ساده ای از شبیه ساز gem5 را نشان می دهد، که نوع داده های قابل دریافت از یک اجرای شبیه ساز را نشان می دهد. درک محیط منحصر به فرد که

[1] Instruction Set Architecture
[2] Docker

یک شبیه ساز در آن عمل می کند، برای طراحی و شبیه سازی سامانه های موثر حائز اهمیت است. این نه تنها بر روند راه اندازی و عملکرد شبیه ساز تأثیر می گذارد، بلکه بر کیفیت و نوع داده های به دست آمده از شبیه ساز نیز تأثیر می گذارد.

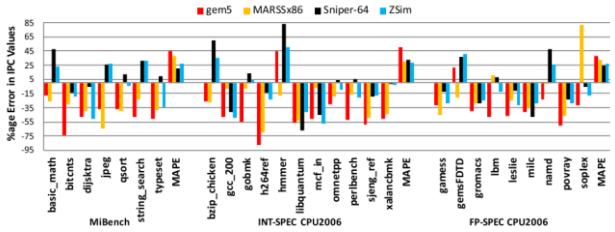
```
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 20.1.0.2
gem5 compiled Nov 20 2020 12:01:27
gem5 started Nov 27 2020 15:22:14
gem5 executing on Dauto98-ROG-Strix-GS31GD, pid 12402
command line: ./build/X86/gem5.opt configs/thesis/X86/se/timing-no-cache-se.py

Global frequency set at 1000000000000 ticks per second
warn: No dot file generated. Please install pydot to generate the dot file and pdf.
warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (512 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
Beginning simulation
info: Entering event queue @ 0.  Starting simulation...
warn: ignoring syscall access(...)
warn: ignoring syscall access(...)
warn: ignoring syscall access(...)
warn: ignoring syscall mprotect(...)
warn: ignoring syscall mprotect(...)
warn: ignoring syscall mprotect(...)
warn: ignoring syscall mprotect(...)
Loop hello 0
Loop hello 1
Loop hello 2
Loop hello 3
Loop hello 4
Loop hello 5
Loop hello 6
Hello hello!
Exiting @ tick 12843465000 because exiting with last active thread context
```

شکل ۷ نمونه‌ای از خروجی شبیه‌ساز gem5 برای برنامه Hello world.

iii. ورودی/خروجی

همانطور که در شکل ۱ نشان داده شده، معمولا ورودی شبیه‌سازها یک سری کانفیگ و پیکربندی اولیه میباشد که با مشخص کردن هر کدام از آنها با توجه به محیط شبیه‌ساز، سیستم مورد نظر در دنیای واقعی شبیه‌سازی خواهد شد. در رابطه با خروجی نیز توقع داریم بعد از اجرای تکه دستوری که میتواند یک قطعه کد اجرای برنامه باشد، به ما خروجی مورد نظر را در زمان مشخص تحویل بدهد. همچنین، در کنار خروجی، توقع داریم که گزارشی جامع در کنار آن از نحوه عملکرد سیستم تا تولید آن خروجی به ما بدهد.

این خروجی در واقع معیار محققان برای مقایسه دو سخت افزار مختلف روی تکه کدی واحد و حالات خاصی از سیستم (مثلا اجرای حلقه‌ها، فرآیندهای موازی و غیره) میباشد. در شکل زیر مقایسه‌ای پیشرفته‌تر در رابطه با ارور IPC در سیستم‌های ۶۴ بیتی در شبیه‌سازهای متنوع آورده شده است.



شکل ۷ درصد خطا در مقادیر IPC برای باینری‌های ۶۴ بیتی

همچنین میدانیم که مثلا برای مدل کردن کش، ابتدا مثلا به شبیه‌ساز میگوییم که حافظه کشی داریم که سایز، مقدار طول هر خانه، زمان دسترسی، تکنولوژی ساخت و غیره را به عنوان کانفیگ ورودی میدهیم و در قبال آن به ازای یک سیاست جایگزاری[1] خاص، تاخیر، تعداد دفعات hit و miss و جزییات بیشتر مشخصات رفتاری را گزارش میکند. مثلا یک کد در حد چند صد خط می‌باشد که یک سری توالی از دسترسی‌های حافظه را می گیرد و از آن گزارش می گیریم و مثلا میخواهیم دو replacement policy را مقایسه کنیم.

به عنوان مثال در شبیه‌ساز deniro خواهیم داشت:

i 0x400400AC 4	Instruction Fetch
i 0x400400B0 4	Instruction Fetch
r 0x40000100 2	Load (Halfword)
i 0x400400B4 4	Instruction Fetch
w 0x40000102 1	Store (Byte)

که به عنوان ورودی به شبیه‌ساز داده میشود و r ، w و i به ترتیب نمادهای Read ، Write و Fetch هستند. و همچنین آدرس داده شده نیز به صورت هگزادسیمال از 0x00000000 تا 0xFFFFFFFF متغیر می‌باشد.

با اجرای این ورودی در شبیه‌ساز و تعریف ساختار حافظه کش، سیستم شروع به کار کردن میکند و در نهایت خروجی اجرا را به صورت یک فایل گزارش درون یک فایل می‌نویسد.

ث. نتیجه گیری

نیازهای خاص برنامه مهمترین عواملی هستند که هنگام کار با یک شبیه ساز باید در نظر گرفت. به عنوان مثال، اگر هدف ما شبیه سازی یک سیستم کامل است که شامل معماری های متنوعی مانند ARM ، X86-64، Alpha و RISC-V است، می توانیم از شبیه ساز gem5 استفاده کنیم. این شبیه ساز چند منظوره است و قادر به نمایش دقیق مدل مفهومی ما است.

با این حال، اگر تمرکز ما فقط بر شبیه سازی یک الگوریتم کش خاص (برای مثال، سیاست های جایگزین LRU یاFIFO) باشد، نیازی به پردازش دستورات سنگین یا صرف زمان و انرژی برای شبیه سازی کل سیستم نداریم. در چنین مواردی، ما می توانیم Deniro را انتخاب کنیم، یک شبیه ساز که به طور خاص منظوره برای کش طراحی شده است.

علاوه بر این ها، MARSS وجود دارد که به طور اختصاصی برای سیستم های X86-64 طراحی شده است. مهم است توجه داشت که هر شبیه ساز نقاط قوت و ضعف خود را دارد و انتخاب شبیه ساز باید با نیازهای خاص برنامه همخوان باشد.

هنگام انتخاب یک شبیه ساز، بسیار حائز اهمیت است که عوامل دیگری مانند منحنی یادگیری مرتبط با شبیه ساز، پشتیبانی جامعه موجود و توانایی شبیه ساز در ارائه نتایج دقیق و قابل اعتماد را نیز در نظر بگیرید. علاوه بر این، قابلیت افزایش مقیاس شبیه ساز نیز باید در نظر گرفته شود، به خصوص برای شبیه سازی های بزرگ. در جدول زیر مقایسه‌ای جامع از تمامی حالات مختلف شبیه‌سازها آمده است[11].

Simulation Model	Accuracy	Performance	Level of details	Easiness of development
Functional simulation	A-	P+++	L	E+++
Timing - cycle accurate simulation	A++	P	L++	E
Timing - event driven simulation	A+	P+	L+	E+
Coupled functional-timing	A+++	P	L++	E
Decoupled functional-timing/ timing first	A+	P	L+	E+
Decoupled functional-timing/ functional first	A	P++	L+	E++
Decoupled functional-timing/ timing direct	A++	P+	L++	E
Full-system simulation	A++	P	L+++	E
User-level simulation	A+	P+	L++	E
Trace-driven simulation	A+	P	L+	E+
executable-driven simulation	A++	P+	L++	E

Note: [evaluation parameter's first letter] with a suffix of +++ represents the highest value and without a suffix represents the lowest value

شکل ۸ مقایسه بین شبیه‌سازها [10].

در نتیجه، درک نیاز های خاص برنامه و تطابق آن ها با قابلیت های شبیه ساز می تواند منجر به شبیه سازی های کارآمد و مؤثر تر شود.

مراجع

[1] S. Borkar, “Cost of Silicon Viewed from VLSI Design Perspective,” in Proceedings of the 2008 International Conference on VLSI Design, pp. 1-6, January 2008.

[2] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, *et al.*, “The gem5 Simulator,” *SIGARCH Comp. Arch. News*, vol. 39, pp. 1–7, May 2011.)

[3] A. Patel, F. Afram, S. Chen and K. Ghose, "MARSS: A full system simulator for multicore x86 CPUs," 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC), San Diego, CA, USA, 2011, pp. 1050-1055.

[4] J. Edler and M. D. Hill. Dinero iv trace-driven uniprocessor cache simulator. http://www.cs.wisc.edu/ markhill/DineroIV/, 2004.

[5] A. Akram and L. Sawalha, "×86 computer architecture simulators: A comparative study," 2016 IEEE 34th International Conference on

[1] Replacement Policy

Computer Design (ICCD), Scottsdale, AZ, USA, 2016, pp. 638-645, doi: 10.1109/ICCD.2016.7753351.

[6] A. Akram and L. Sawalha, "A Survey of Computer Architecture Simulation Techniques and Tools," in IEEE Access, vol. 7, pp. 78120-78145, 2019, doi: 10.1109/ACCESS.2019.2917698.

[7] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli. "Accuracy evaluation of GEM5 simulator system". In: *7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*. 2012, pp. 1–7. DOI: 10.1109/ReCoSoC.2012.6322869.

[8] Jose Luis Bismarck Fuentes Morales. "Evaluating Gem5 and QEMU Virtual Platforms for ARM Multicore Architectures". MA thesis. KTH, School of Information and Communication Technology (ICT), 2016, p. 63.

[9] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli, "Accuracy Evaluation of GEM5 Simulator System," in Int. Workshop on Reconfigurable Communication-centric Systems-onChip, pp. 1–7, 2012.

[10] https://www.gem5.org/getting_started/

[11] A. Akram and L. Sawalha, "A Survey of Computer Architecture Simulation Techniques and Tools," Department of Computer Science, University of California at Davis, Davis, CA 95616, USA and Department of Electrical and Computer Engineering, Western Michigan University, Kalamazoo, MI 49008, USA.