Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 10, Number 2, Summer 2021, pp. 68-81
Received: 12/22/2020, Revised: 05/03/2021, Accepted: 06/22/2021

Investigation of the Effect of Burst Multi-bit Soft Errors on Control Flow and Data Error Behaviors of Embedded Systems

Moona Yakhchi[1], Mahdi Fazeli [2,1*] and Seyed Amir Asghari Toochai[3,1]
1- Department of Computer, Borujerd Branch, Islamic Azad University, Borujerd, Iran.
2*- Department of Computer Engineering, Bogazici University, Istanbul, Turkey.
3- Electrical and Computer Engineering Department, Kharazmi University, Tehran, Iran.
[1] M.yakhchi@iaub.ac.ir, [2,1*] Mahdi.fazeli@boun.edu.tr, and [3,1] asghari@khu.ac.ir

Corresponding author's address: Mahdi Fazeli, Faculty of Computer Engineering, Bogazici University, Istanbul, Turkey.

Abstract- Soft errors caused by high energetic particles have become a serious threat for reliability of today's computer systems. Until recently, Single-Bit soft errors has considered as the main effect of particle strikes, however, as technology is downscaling toward Nano-Scale sizes, Multi-Bit soft errors is emerging as an important reliability challenge. Consequently, investigating the effects of Multi-Bit soft errors on systems reliability are of decisive importance. In this paper, we intend to comprehensively study the effects of Multi-Bit soft errors on system behaviors in terms of Control Flow and Data Errors using different benchmarks. According to our fault injection experiments using 17 MiBench programs and injected 17000 fault, observed Multi-Bit faults have same behavior like Single-Bit faults and rate of the fault have significantly increased. Therefore, the average SDC rate has increased by 2% in multi-bit errors, which is due to the non-symptomatic nature of these errors. This indicates a negligible impact on system reliability.

Keywords- Multi-Bit fault, Single-Bit fault, Fault injection, Control Flow Error, Data Error, Soft Error.

مجله علمی پژوهشی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۱۰، شماره ۲، تابستان ۱۴۰۰، صفحه ۶۸-۸۱
دریافت: ۱۳۹۹/۱۰/۰۲، بازنگری: ۱۴۰۰/۰۲/۱۳، پذیرش: ۱۴۰۰/۰۴/۰۱

بررسی تأثیر خطاهای نرم چند بیتی قطاری بر رفتارهای جریان کنترل و داده در سیستم‌های نهفته

مونا یخچی[۱]، مهدی فاضلی*[۲و۱]، سید امیر اصغری توچائی[۳و۱]
۱- گروه کامپیوتر، واحد بروجرد، دانشگاه آزاد اسلامی، بروجرد، ایران.
*۲- دانشکده مهندسی کامپیوتر، دانشگاه بغازیچی، استانبول، ترکیه.
۳- دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی خوارزمی، تهران ، ایران.
[1]m.yakhchi@iaub.ac.ir, [2,1*]Mahdi.fazeli@boun.edu.tr, [3,1]asghari@khu.ac.ir

* نشانی نویسنده مسئول: مهدی فاضلی ، ترکیه، استانبول،دانشگاه بغازیچی، دانشکده مهندسی برق و کامپیوتر.

چکیده- امروزه، خطاهای نرم که ناشی از برخورد ذرات پرانرژی است به تهدیدی جدی برای قابلیت اطمینان سیستم‌های کامپیوتری تبدیل شده است. در طول سال‌های اخیر ، خطاهای نرم تک بیتی به عنوان اصلی‌ترین تأثیر حملات ذرات در نظر گرفته شده‌اند. از آنجایی که تکنولوژی به سمت ابعاد نانومتری پیش می‌رود و با توجه به این واقعیت که نرخ رخداد اشکالات چندبیتی قابل مقایسه با خطاهای تک بیتی است. لذا خطاهای نرم چند بیت به‌عنوان یک چالش مهم و تاثیرگذار بر روی قابلیت اطمینان ظاهر می‌شوند. در نتیجه، بررسی اثرات خطاهای نرم چند بیتی بر سیستم‌های کامپیوتری از اهمیت اساسی برخوردار است. در این مقاله، اثرات خطاهای نرم چند بیتی بر رفتارهای سیستم از نقطه نظر خطاهای جریان کنترل و خطاهای داده به طور جامع با استفاده از برنامه های محک مختلف مورد بررسی قرار گرفته است. طبق آزمایشات تزریق اشکال که با استفاده از ۱۷ برنامه محک MiBench و تزریق ۱۷۰۰۰ اشکال صورت گرفته است، اشکالات چند بیتی مشاهده شده رفتار مشابهی با اشکالات تک بیتی را نشان داده‌اند، با این تفاوت که به طور میانگین میزان SDC، در خطاهای چند بیتی ۲ درصد افزایش یافته است که به دلیل ماهیت بدون نشانه بودن این دسته از خطاها، تاثیر آنها بر روی قابلیت اطمینان قابل توجه است .

واژه‌های کلیدی: اشکال چند بیتی، اشکال تک بیتی، تزرق اشکال، خطای جریان کنترل ، خطای داده، خطای نرم.

۱- مقدمه

امروزه، کاهش اندازه تراشه‌ها، افزایش تعداد ترانزیستور به ازای هر تراشه، همراه با کاهش سطح ولتاژ و همچنین افزایش فرکانس سیگنال ساعت، منجر به رشد نمایی نرخ خطای نرم[۱] مدارهای دیجیتال شده است [۱-۳]. با توجه به این واقعیت که نرخ رخداد اشکالات چند بیتی در مقیاس‌های نانو با خطاهای تکی قابل مقایسه شده است [۴و۵]، یک طرح مناسب برای شناسایی رفتار آن‌ها مورد نیاز است. زیرا خطاهای نرم چند بیتی نقش بسزایی در به چالش کشیدن قابلیت اطمینان سیستم‌های کامپیوتری دارند[۶-۱۱]

خطاهای نرم را می توان در سطح سیستم به دو دسته طبقه‌بندی کرد[۱۲]: ۱) خطای جریان کنترل (CFE)[۲]، که به هرگونه انحراف از جریان کنترل صحیح یک برنامه اشاره دارد. ۲) خطای داده، این خطاها به هر گونه دستکاری داده‌ها ناشی از خطاهای گذرا اشاره دارد. اگر داده‌ها محافظت نشوند، چنین خطاهایی ممکن است منجر به خرابی سیستم شود. این دو کلاس از خطاها را می توان بر اساس رفتار آنها در چهار دسته زیر همانند [۱۳] طبقه بندی کرد:

خوش خیم/ پوشیده شده[۳]: این خطاها هیچ تاثیری در نتیجه برنامه ندارند. آن‌ها ممکن است در طول اجرای برنامه پوشانده شوند یا راهی برای تأثیر بر نتایج خروجی پیدا نکنند [۶][۱۳].

خرابی/ قطع[۴]: هنگامی که یک خطای نرم باعث می‌شود سیستم وارد حالتی شود که در آن به هیچ رویدادی پاسخ ندهد، از آن به عنوان وضعیت خرابی/ قطع یاد می‌شود [۶][۱۳].

خرابی آرام داده (SDC)[۵]: منجر به نتیجه خروجی نادرست یا اشتباه می‌شود. آن‌ها را خطاهای خاموش می‌نامند زیرا مکانیزم-های محافظتی نمی‌توانند آن‌ها را تشخیص دهند. در واقع، این نوع خطا تهدیدی جدی برای قابلیت اطمینان سیستم است و محافظت از سیستم در برابر آن‌ها سخت است [۶][۱۳].

نقض زمانی[۶]: هنگامی که یک سیستم معیوب در یک زمان خاص نتیجه ای ایجاد نمی‌کند. به عنوان یک خطای نقض زمانی در نظر گرفته می‌شود [۱۲].

برای طراحی مکانیسم‌های حفاظتی کارآمد، توصیف سهم هر یک از کلاس‌های خطای ذکر شده در خرابی‌های سیستم بسیار مهم و اثر گذار است. برای محافظت سیستم در مقابل چنین اشکالاتی نیاز به شناسایی این اشکالات است. بنابراین، از تزریق اشکال برای شبیه‌سازی رفتار سیستم در صورت وجود خطای نرم استفاده می-شود. ابزار تزریق اشکال نرم‌افزاری به دلیل سریع‌تر بودن از ابزار سخت‌افزاری و عدم نیاز به سخت‌افزار اضافی کاربرد گسترده‌ای دارد [۶].

یکی از چالش‌های مهم در تزریق اشکال، مدل اشکال است. پیش از این مدل اشکال تک بیتی یک مدل محبوب شناخته می‌شد و در بسیاری از تحقیقات مورد استفاده قرار می‌گرفت. ولیکن، مطالعات اخیر نشان داده است که منشاء بسیاری از خطاها اشکالات چند بیتی است [۶][۹][۱۴-۱۵]، از اینرو در مدل‌های اشکال هر دو اشکال تک و چند بیتی باید در نظر گرفته شوند. تحقیقات دهه‌های گذشته ابزارهای مختلف تزریق اشکال برای ارزیابی قابلیت اطمینان و آزمون نرم افزار ارائه کرده‌اند. این ابزارها شامل روش‌های مختلف تزریق اشکال اعم از پیاده‌سازی سخت-افزاری، نرم‌افزاری، تقلید و شبیه‌سازی است [۱۶-۲۰]. در بسیاری از تحقیقات انجام شده تکیه بر بررسی اشکالات تک بیتی شده است. کار بسیار کمی در زمینه مطالعه اثرات اشکالات چندبیتی بیت فراتر از دو بیتی وجود دارد [۶] [۲۱-۲۳] حتی آن‌ها فقط داده‌هایی را که در دستورالعمل‌های بارگذاری و ذخیره استفاده می‌شود در نظر گرفته‌اند و در [۲۴] به بررسی خطا های چند بیتی به صورت تک بیتی و تنها رفتار سیستم بر روی SDC پرداخته اند.در [۱۱] به ارائه روشی جهت تشخیص و تصحیح خطای دو و سه بیتی با روشی به نام دو بیت توازن و یک افزونه تنها در پردازنده‌های ARMv7 پرداخته است. [۱۰] به بررسی و تحلیل عناصر پردازنده ARM Cortex-A9 این کار بر روی فضای محدود، مجاور و تا سه بیت اشکال صورت گرفته است. برخی از پژوهش-های صورت گرفته بر روی ارائه روشی برای مقابله با برخی اثرات اشکالات تک بیتی تمر کز کرده‌اند. در [۲۵-۲۹] روشی برای مقابله با اثرات خطاهای نرم ارائه کرده‌اند. [۳۰-۳۲] روی ارائه روش برای مقابله با خطاهای جریان کنترل کار شده است. مطالعات [۳۳-۳۶][۱۳] برای مقابله با خطاهای داده روش‌هایی ارائه داده‌اند [۱۳][۳۳]. [۱۳][۳۳][۳۶][۴۶-۳۸] نیز از تکنیک ها هوش مصنوعی و یادگیری ماشین برای شناسایی دستورات مستعد SDC و یا دسته بندی دستورات استفاده کرده اند. علاوه بر اینکه همه این مطالعات تنها به بررسی خطاهای تک بیتی و برخی دستورات پرداخته اند. مشکل اصلی در این کارها این است که تنها اشکالات تک بیتی در نظر گرفته شده است. همچنین حساسیت متفاوت دستورالعمل‌ها نسبت به خطاهای تک و چند بیتی را نادیده گرفته‌اند.

بنابراین، هدف این مقاله بررسی رفتاراشکالات تک و چند بیتی بدون محدودیت در دستورالعمل ها و طبقه بندی این رفتارها است. سوال اصلی این تحقیق این است که اثر خطاهای چند بیتی بر روی نرم افزارها چگونه است؟ و با اشکالات تک بیتی تا چه اندازه متفاوت است؟ برای این کار، ابزار تزریق خطای نرم‌افزاری در پایتون[۷] پیاده‌سازی شد و یک پایگاه داده در Mysql ایجاد شد تا رفتار اشکالات تک و چند بیتی را در ۱۷ محک بررسی شود. علاوه بر این، اثر خطا در تمام دستورالعمل‌ها و داده‌های موجود در برنامه مورد مطالعه قرار گرفت. ۱۷ محک انتخاب شد تا یک مطالعه کلی صورت پذیرد.

ادامه مقاله به شرح زیر است: بخش ۲ اصطلاحات را معرفی می کند، بخش ۳ آثار مرتبط را نشان می‌دهد. شبیه‌سازی، محیط و ابزار تزریق اشکال در بخش ۴ شرح داده شده است. بخش ۵ نتایج بدست آمده را پس از تزریق اشکال ارائه و مورد بحث قرار می‌دهد. سرانجام، نتیجه‌گیری مقاله در بخش ۶ ارائه شده است.

۲- اصطلاحات و تعاریف

خطای نرم: کاهش اندازه ابعاد در تکنولوژی ساخت مدارات مجتمع، افزایش نویز و تراکم تعداد قطعات موجود در فضا، باعث شده است ذرات پرانرژی در پرتوهای کیهانی در سامانه‌های

کامپیوتری اشکالاتی تولید کنند که به خطاهای نرم معروف هستند [۳۱]. علت اینکه لفظ نرم برای این خطاها استفاده می‌شود این است که اثر این خطاها به شکل خرابی یک قطعه نمی باشد بلکه به صورت تغییر در محتوای عناصر حافظه است که مجددا با بازنویسی در حافظه اصلاح می‌شوند. در حقیقت خطاهای نرم در مقابل خطاهای سخت هستند که در آن‌ها خطا منجر به خرابی کامل قطعه می‌شود و برای بازیابی سامانه باید فرآیند تعمیر یا تعویض انجام شود. تغییر در محتوای حافظه به‌دلیل رخداد این خطاها می‌تواند به شکل یک اشکال تک بیتی یا یک اشکال چند بیتی [۳۷][۴۷] باشد. در اشکال تک بیتی فقط یک بیت حافظه دچار تغییر می‌شود حال آنکه در اشکال‌های چند بیتی چند بیت مجاور در حافظه دچار تغییر می‌گردند [۴۸]. خطای نرم در دو دسته اصلی بر روی نرم افزار در حال اجرا تأثیر می گذارد: خطای جریان کنترل و خطای داده.

خطای جریان کنترل: هنگامی رخ می‌دهد که پردازنده دستورالعمل‌های غیرمنتظره‌ای را اجرا می‌کند و سپس آن را از جریان صحیح برنامه منحرف می‌کند. بنابراین، برنامه به یک مکان نادرست پرش می‌کند و توالی اجرا را تغییر می‌دهد که ممکن است منجر به رفتار غیرقابل پیش‌بینی سیستم شود. نتایج سیستم به عنوان خوش بین/ پوشیده شده، خرابی/ قطع، SDC و مهلت‌زمانی طبقه‌بندی شده است که در بخش قبلی شرح داده شد. برای مدل‌سازی خطاهای جریان کنترلی، لازم است اثرات این خطاها بر روی رفتار دستورالعمل‌های برنامه بررسی شود. طبق تحقیقات موجود در این حوزه، رفتار خطاهای نرم رخداده در بخش دستورالعمل‌های یک برنامه به سه دسته تقسیم می شوند [۳۰]:

درج پرش[۸] : خطا ممکن است منجر به تغییر دستورالعمل غیر پرشی به دستور پرشی و در نتیجه باعث بروز خطا شود [۳۰].

حذف پرش[۹] : خطا ممکن است منجر به تغییر دستورالعمل پرش به دستور غیر پرشی و در نتیجه باعث بروز خطا شود [۳۰].

تغییر مقصد پرش[۱۰] : خطا ممکن است منجر به تغییر در آدرس هدف دستورالعمل پرش و در نتیجه باعث بروز خطا شود [۳۰].

خطای داده: این خطا انحراف از داده‌های صحیح است هنگامی که یک اشکال تک بیتی یا چند بیتی در زمان پردازش داده‌ها اتفاق می‌افتد. بعبارت دیگر این خطا در داده آشکار می‌شود. نتایج سیستم به عنوان خوش بین/ پوشیده شده ، خرابی/ قطع، SDC و نقض زمانی طبقه‌بندی می‌شوند که در بخش قبلی و CFE شرح داده شده‌اند.

تزریق خطا: برای بررسی رفتار سیستم در حضور اشکال نیاز به تزریق اشکال است. روش‌های مختلفی برای تزریق اشکال ارائه شده است که به ۴ دسته کلی، سخت افزاری، نرم افزاری، شبیه سازی[۱۱] و تقلید[۱۲] تقسیم می‌گردد. قرار دادن یک سیستم در فضایی که مورد تشعشع و پرتوهای کیهانی باشد زمان و هزینه زیادی می‌برد لذا در جهت داشتن اطلاعات سیستم در این حالت اقدام به شبیه‌سازی اشکالات با استفاده از ابزارهای تزریق اشکال می‌شود. تزریق اشکال روشی برای معرفی اشکال در حالت قاعده‌مند مدیریت کنترل و مطالعه بر روی رفتار سیستم است. تزریق اشکال روشی در تقلید اثرات خرابی اشکالات سخت‌افزاری به‌روی نرم‌افزار با مختل کردن مقادیر داده/دستورات انتخاب شده در برنامه است. محدودیت اصلی تزریق اشکال این است که به‌دست آوردن پوشش و نمایش کامل می‌تواند بسیار سخت باشد. از سوی دیگر پیگیری یک اشکال تا حصول اثر آن در سیستم نیز دشوار است. بنابراین، از روش‌های نرم‌افزاری تزریق اشکال به‌دلیل ارزان‌تر بودن، عدم نیاز به سخت افزار اضافی، ارائه سطح بالایی از کنترل و امکان اجرای چندباره استفاده شده است [۴۹]. باید در نظر داشت که تزریق اشکال می‌تواند در سطوح ریز معماری و زبان انتقال ثبات هم صورت گیرد. اما، این روش‌ها مقیاس‌پذیر نیستند چون باید از جزئیات ریز معماری‌ها و سطح ثبات انتقال (RTL)[۱۳] مطلع بود. به‌همین دلیل، تزریق اشکال در سطوح بالاتر مثل اسمبلی انجام می‌شود. هدف به‌دست آوردن پوشش کافی از نظر تعداد دستورات اجرا شده نسبت به حالت سخت‌افزاری تحت تزریق است. با این اوصاف در این پژوهش تزریق اشکال نرم‌افزاری در نظر گرفته شده است. که اشکال به‌صورت تک بیتی وچند بیتی به برنامه تزریق می‌شود.

۳- کارهای مرتبط پیشین

تحقیقات گسترده‌ای بر روی ارائه روش‌های مقابله با خطاهای نرم انجام شده است. اما تعداد معدودی به بررسی رفتار سیستم در حضور اشکالات چند بیتی پرداخته‌اند. بررسی سیستم همان‌گونه که در بخش قبلی تشریح شد نیازمند تزریق اشکال است. ازیک سو تحقیقات پیشین نشان می‌دهند که ویژگی‌های مختلفی از برنامه بر روی نتایج خروجی اشکالات تاثیر گذار است به عنوان مثال نحوه پیاده‌سازی برنامه، سطوح مختلف بهینه‌سازی کامپایلری و تعداد مقادیری ورودی [۲۱]. از سوی دیگر فاکتورهای بسیاری بر روی تعداد SDC اثر می‌گذارد که شامل تعداد متغیرها، تعداد دستورات پویا، اندازه کد و محتویات ثبات‌ها یا حافظه و نوع دستورات در حال اجرا می‌باشند [۲۲]. اما در مقاله پیش‌رو رویکرد

کلی برای بررسی رفتار سیستم فارغ از ویژگی‌های برنامه و یک خروجی خاص است. لذا در بررسی کارهای پیشین در ابتدا به مقالاتی که رفتارهای اشکالات تک بیتی و چند بیتی را بررسی کرده‌اند پرداخته شده است. سپس برخی ابزارهای تزریق اشکال نرم‌افزاری بررسی شده‌اند.

در [۲۱] نویسنده اثر خطاهای تک بیتی و دوبیتی را بر روی ثبات‌ها و حافظه اصلی بررسی کرده است. که در آن به تقلید رفتار سخت افزاری‌خطا درسطح اسمبلی پرداخته است. خطاهای دوبیتی به صورت دوبیت جدا در یک کلمه صورت می‌گیرد. همچنین در این پژوهش حساسیت به خطا برای مکان‌های مختلف مورد بررسی قرار گرفته است. نتایج تزریق اشکال نشان می‌دهند خطاهای دوبیتی و تک بیتی از لحاظ درصد رخداد SDC تقریبا یکسان هستند. همچنین نرخ بالاتر خرابی را در خطاهای دوبیتی اعلام نموده‌اند. نویسنده معتقد است هر چه بیت بیشتری اشکال تزریق شود سهم خرابی بالاتر می‌رود. تفاوت کار تحقیق پیش رو با آنها در فرض مدل تزریق اشکال است. اولاً که در این تحقیق به بررسی خطاهای ۱ تا ۳ بیت پرداخته شده است. ثانیاً خطاهای چند بیتی به صورت قطاری در یک مکان بررسی شده است. زیرا احتمال رخداد خطا در دو دستور متفاوت طبق [۵۰] ۱ به دو میلیارد است. لذا خطاهای قطاری را مد نظر قرار داده شده است یعنی یک خطای چند بیت منجر به تغییر چند بیت مجاور یکدیگر شود.

در [۲۳] به بررسی اثر خطاهای تک‌بیتی در یک مکان ثبات به صورت تصادفی پرداخته شده است. این کار در سطح کدمیانی تولید شده با LLVM انجام گرفته است. به این صورت که دو مکان تصادفی انتخاب و یک بیت اشکال به هر مکان تزریق می‌شود. سپس به بررسی آسیب‌پذیری نرم‌افزاری با توجه به این اشکالات پرداخته شده است. در نهایت نتایج نشان داده است که در رفتار دو خطای تک بیتی[۱۴] مقادیر خرابی بسیار بالاتر از تک بیت است و SDC کمتر و دو خطای تک بیتی رفتار مختلفی در برنامه‌های متفاوت نشان می‌دهند. تفاوت کار آن‌ها با این تحقیق در نحوه تزریق اشکال است که به صورت تک بیتی است و اینکه تنها تزریق اشکال به ثبات‌ها را درنظر گرفته‌اند. از سوی دیگر سطح انتزاع به کار رفته آنها با این تحقیق متفاوت است.

بهروز سنگ شعله‌ای و همکارانش نیز مطالعه‌ای بر روی خطاهای چند بیتی در یک کلمه یا در کلمات متفاوت انجام داده‌اند. تزریق اشکال چند بیت تا ۳۰ خطا درکلمه یا کلمات متفاوت انجام شده به این صورت که خطا به صورت تک بیت در بررسی چند بیت به برنامه تزریق می‌گردد، تا زمانی که به حداکثر تعداد خطای مورد نظر برسند یا اینکه سیستم متحمل خرابی شود. این کار نیز در سطح کد میانی بواسطه کامپایلر LLVM و در ازای محاسبه درصد رخداد SDC در خطاهای یک و چند بیت صورت گرفته است. آن-ها روشی برای هرس کردن مکان‌های تزریق اشکال نیز ارائه داده-اند زیرا نویسندگان معتقدند که روش‌های مختلف هرس کردن در تزریق اشکال تک‌بیتی قابلیت گسترش در مدل‌های تزریق اشکال چند بیت را ندارند. در نهایت نتایج تزریق نشان داده است که فاصله دستورات بین یک تزریق تا تزریق بعدی در خروجی اثر ندارد. مقدار SDC رخداده در چند بیت کمتر از تک بیت است ولی در ۸ درصد از کمپین‌ها درصد SDC بیشتری در چند بیت مشاهده شده است. همچنین برای اغلب کمپین‌ها در صد بالای SDC در خطای سه بیت است. تفاوت آن‌ها با این تحقیق در مدل تزریق اشکال است. در این تحقیق اشکالات به صورت قطاری یعنی بر روی بیت‌های مجاور در نظر گرفته شده است. این به دلیل گستردگی فضای خطاهای چند بیتی است [۶].

سنگ شعله‌ای در [۲۴] به بررسی رفتار سیستم در برابر خطاهای یک و چند بیتی از نقطه نظر SDC پرداخته است. در این کار از یک تکنیک هرس مبتنی بر خوشه‌بندی خطا با هدف کاهش تعداد تزریق اشکال به مکان‌هایی که احتمال تبدیل به SDC دارند استفاده کرده است. در این روش هرس از یک معیار بیشترین تعداد خطا تا بیسترین تعداد دستور پویا پس از تزریق خطا پیروی می‌کند.تمرکز این مقاله بررسی اثر اشکالات چند بیتی بر روی SDC است و در این حین به هرس مکان‌هایی که اثر کمتری در SDC دارند و یا منجر به شکست زود هنگام سیستم می‌شوند، پرداخته شده است و دستوراتی که منجر به شکست زود هنگام می‌شوند مشخص و سعی به بررسی بیشتر تا هرس آنها کرده است. درصد تزریق اشکال چند بیتی در این مقاله مشخص نیست و اشکالات چند بیتی به صورت تک بیتی در یک بازه زمانی تا پایان حد آستانه و یا شکست سیستم صورت گرفته است.

[۱۱] به ارائه روشی جهت تشخیص و تصحیح خطای دو و سه بیتی با روشی به نام دو بیت توازن و یک افزونه پرداخته است. این روش به خوبی یک افزونه سه تایی[۱۵] نیست و یک روش متعادل برای تشخیص خطای یک تا سه بیتی را دارد. تمرکز نویسنده بر روی ایجاد یک تعادل بین تشخیص و تصحیح و هزینه هایی مثل سربار زمانی، حافظه و کارایی است. تکنیک ارائه شده فعلاً بر روی شبیه‌سازی ARMv7 است.

[۱۰] به بررسی و تحلیل عناصر پردازنده ARM Cortex-A9 که با استفاده از شبیه ساز ریز معماری Gem5 ساخته شده است، از

لحاظ رفتار مقابل اشکالات تک و چند بیتی پرداخته شده است. در این بررسی حافظه نهان L1D بیشترین آسیب پذیری در مقابل خطاهای چند بیتی را دارد. تزریق اشکال در این کار بر روی فضای محدود، مجاور و تا سه بیت اشکال صورت گرفته است و معیار مورد بررسی میزان آسیب پذیری بوده است.

SEInjector یک ابزار تزریق اشکال پویا نرم‌افزاری است که با هرس مکان اشکال تعداد تزریق اشکال را به شرح زیر بهینه می-کند: اول، ثبات استفاده نشده ممکن است اشکال‌ها خرابی یا نقض زمانی شوند مثل اشکال در بیت بالای ثبات RIP. دوم، خطاها را با رفتار مشابه در یک کلاس معادل قرار داده و از هر کلاس یک خطا انتخاب می‌کند. مدل خطا در این ابزارها بر اساس رویداد تکی است و یک محدوده اشکال را در پرونده ثبات در نظر می‌گیرد و از چارچوب پین[16] استفاده می‌کند. استراتژی تزریق خطای آنها، خطا را در هر اجرا و در سطح قطعه کد تزریق می‌کند. آن‌ها نتایج را به چهار دسته خروجی صحیح، SDC ، مهلت‌زمانی و استثنا تقسیم می‌کنند [۱۶-۱۷].

چانگ و همکاران Hamartia را پیشنهاد کرده‌اند، یک ابزار دقیق دودویی باینری مبتنی بر چارچوب سلسله مراتبی که تزریق در سطح اسمبلی تا سطح دروازه[17] را در نظر دارد و در آن مانند [۱۷-۱۶] از پین استفاده شده است. یک سطح دروازه اضافی در هر گره برای اتصال تزریق به ثبات انتقال وارد شده و توسط Pyverilog انجام شده است. آنها روی X86 کار کرده‌اند. مدل آن‌ها مبتنی بر اشکال تک بیتی، اشکال دو بیتی و سطح دروازه RTL بوده است. RTL منجر به خطاهایی متفاوت از اشکال تک‌بیتی در سطح دستورالعمل شده است. این ابزار خطا را فقط به واحدهای حسابی تزریق می‌کند. نتیجه تزریق به پوشیده شده، خطای شناسایی شده غیر قابل تصحیح وSDC تقسیم شده است [۱۸].

GOOFI یک ابزار تزریق اشکال شی‌گرا است. GOOFI می‌تواند اشکال را به سیستم‌های مختلف هدف تزریق کند و کاربر می‌تواند با استفاده از رابط کاربری گرافیکی این ابزارها، روش جدید تزریق اشکال را به سیستم هدف اضافه کند. GOOFI به زبان Java نوشته شده است. اطلاعات در طول آزمون در یک پایگاه داده SQL جمع آوری می‌شود. این قابلیت تزریق اشکال نرم‌افزاری، اشکال را به داده‌ها و دستورالعمل‌ها قبل از زمان اجرا تزرق می-کند. تزریق اشکال پیاده‌سازی شده زنجیره اسکن به حالت داخلی تزریق می‌کند [۱۹].

GOOFI-2 نسخه پیشرفته GOOFI قبلی با برخی روش‌های اضافی است. GOOFI-2 با استفاده از درگاه‌های دسترسی آزمایشی سازگار با Nexus، اشکال را در میکروکنترلرهای Freescale565 MPC و MPC5554 تزریق می‌کند. GOOFI و GOOFI-2 از اشکال‌های تک‌بیتی و چندگانه به صورت تک بیت پشتیبانی می-کنند [۲۰].

آنا توماس و همکارانش ابزاری برای تزریق خطای مبتنی بر LLVM به نام LLFI ساختند. تزریق اشکال در ماشین مجازی سطح پایین ساخته شده است که مجموعه‌ای از ابزارها و اجزای کامپایلر است و امکان بهینه‌سازی کد را می‌دهد. آن‌ها اشکال را بر دستورات بارگذاری و ذخیره کردن در سطح میانی تزریق کرده‌اند. این ابزار تزریق اشکال سطح بالا بر اساس کامپایلر LLVM و سازوکار تزریق اشکال نرم‌افزاری ساخته شده است. LLFI اشکالی را که توسط برنامه و داده خوانده می‌شود در نظر می‌گیرد و خطایی را که ممکن است با سخت افزار پوشیده شود در نظر نمی‌گیرد. برای مقایسه دقت LLFI، نویسندگان ابزاری برای تزریق اشکال در سطح اسمبلی به نام PINFI ساخته‌اند. PINFI با استفاده از ابزار PIN از Intel اشکال را تزریق می‌کند. PIN یک ابزار دودویی پویا است که می‌تواند رفتار باینری x86 را تغییر و ردیابی کند. این ابزار سطح پایین است. سرانجام، با مقایسه نتیجه تزریق LLFI و PINFI مشاهده کرده‌اند که LLFI از PINFI در نرخ SDC و PINFI از LLFI در نرخ خرابی دقیق‌تر است [۵۱].

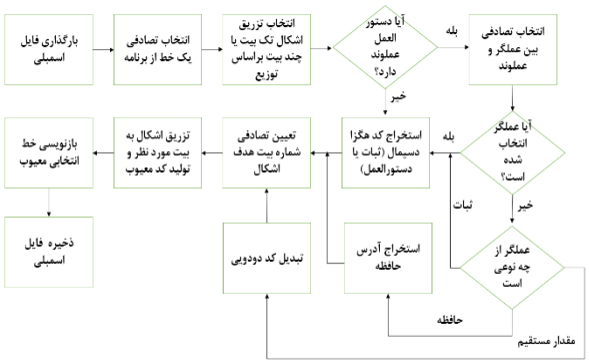
۴- روش کار

۴-۱- مدل و پیاده‌سازی ابزار تزریق اشکال

ضرورت بررسی اثر انواع اشکال در یک برنامه در اختیار داشتن اطلاعات کاملی از نوع و محل رخداد اشکال و نحوه رفتار برنامه پس از آن است. این اطلاعات از دو راه در دسترس خواهد بود: ۱) جمع‌آوری اطلاعات اشکالات و برنامه در محیط واقعی، ۲) شبیه‌سازی اشکالات و تزریق هدفمند و تحت کنترل آن به برنامه. موانع و محدودیت‌هایی در جمع‌آوری اطلاعات محیط واقعی وجود دارد، بعنوان مثال ممکن است در اجرای واقعی از هر ۱۰۰ اجرا بخشی از برنامه تنها یک بار اجرا گردد. با توجه به این محدودیت و همچنین دیگر امکاناتی که شبیه‌سازی ایجاد می‌کند، نظیر قابلیت دنبال کردن اشکال و اثرات آن و امکان تزریق هدفمند به هر قسمتی از برنامه، اغلب در مطالعات علمی راه دوم انتخاب می‌شود. برای ارزیابی رفتار سیستم‌ها در برابر خطاهای تک بیتی و چند بیتی در داده‌ها و دستورالعمل‌ها ، ابزاری جدید برای تزریق اشکال در محیط پایتون و پایگاه‌داده MySQL طراحی و پیاده-سازی شده است. در ابزار تولیدی اشکالات به هر دو بخش عملوند

و عملگر دستورالعمل تزریق می‌شود. این اشکالات بصورت تک بیتی و چند بیتی قطاری (۲ تا ۳ بیت) صورت می‌گیرد. توزیع تنوع خطاهای تزریقی بر اساس نتایج محققان [۵۳ و ۵۲] در مورد تعداد و چگونگی اشکالات چند بیتی در حافظه ایستا ۴۵ نانومتر صورت می‌گیرد. نحوه انتخاب تعداد بیت و محل اصابت (به عملگر یا عملوند کدام دستور) اشکال به صورت تصادفی و با استفاده از توزیع یکنواخت است.

در اغلب مطالعات صورت گرفته پیشین اشکالات بر روی کد میانی یا کد اسمبلی تزریق شده است. همچنین در اکثریت موارد از ابزار LLFI که سازگار با کامپایلر LLVM است جهت تزریق اشکال در کد میانی استفاده شده است. در این مقاله هم از کامپایلر LLVM استفاده شده است. اما با توجه به نزدیک‌تر بودن کد اسمبلی به واقعیت و توانایی LLVM در تولید کد اسمبلی علاوه بر تولید کد میانی، در این مقاله، تصمیم به ارائه یک ابزار تزریق اشکال است که می‌تواند اشکال تک بیتی و چند بیتی در سطح اسمبلی تزریق کند. برای این منظور، از Clang برای تولید کد اسمبلی استفاده شده است. لازم به ذکر است که در هر بار تزریق اشکال تنها یک اشکال، یک بیتی یا چند بیتی قطاری، تزریق می‌گردد. در هر مرحله از تزریق اشکال، به احتمال ۵۲ درصد تزریق اشکال تک بیتی و با احتمال ۴۸ درصد تزریق اشکال چند بیتی، ۲ تا ۳بیت بر اساس [۵۳ و ۵۲]، طبق شکل ۱ و به صورت قطاری انجام می‌شود. همانگونه که اشاره شد جهت تزریق اشکال می‌بایست کدهای هگزادسیمال دستورالعمل و ثبات در دسترس باشند. این اطلاعات بر اساس [۵۴] در یک پایگاه داده در MySQL ذخیره و مورد استفاده قرارمی‌گیرد. این اطلاعات شامل دستورالعمل‌ها و عملوندهای مربوطه و کد هگزادسیمال دستورالعمل‌ها می‌باشد. دستورالعمل‌های ۶۴ بیتی و ۳۲ بیتی در کد هگزادسیمال متفاوت هستند. از سوی دیگر، برخی دستورالعمل‌ها دارای قابلیت عملوندهای ۸ ، ۱۶ ، ۳۲ و ۶۴ بیتی هستند، که کد هگزادسیمال متفاوتی دارند. برای مثال، کد هگزادسیمال دستورالعمل Mov متفاوت است و شامل Mov، Movd، Movl و Movq که کدهای 0x1، 0x2، 0x3 و غیره همه مربوط به دستورالعمل Mov هستند. این پایگاه‌داده در MySql جمع‌آوری شده است.



شکل۱: روندنما مراحل تزریق اشکال.

روند کلی تزریق اشکال به شرح شکل ۱ است. هر دستورالعمل از دو قسمت تشکیل شده است: یک بخش کد و قسمت دیگر داده. تزریق بیت‌ها در بخش داده‌ها به سه دسته تقسیم می‌شود. داده-های مستقیم، آدرس‌های داده ذخیره شده در حافظه یا ثبات و مقادیر داده‌های ذخیره شده در ثبات‌ها. عملوندها نیز به چندین دسته تقسیم می‌شوند. ابتدا داده‌های مستقیم که به صورت اعداد هستند، به عنوان مثال MOV ax,$10، را به داده‌های باینری تبدیل می‌کند و سپس به صورت تصادفی محل تزریق اشکال انتخاب می‌شود. در خصوص تزریق اشکال به ثبات‌ها به این دلیل که تحت محافظت مکانیزم‌هایی همچون کد تصحیح خطا و بیت توازن هستند امکان تزریق اشکال به محتوای آنها وجود ندارد. ولی ممکن است یک ثبات به ثبات دیگر تبدیل شود. به عبارت دیگر، همانگونه که اشاره شد هر ثبات دارای کدی منحصر بفرد است که در پایگاه‌داده ذخیره شده است. با تزریق اشکال ممکن است کد یک ثبات به کد ثبات دیگر تبدیل شود. به هر برنامه محک ۱۰۰۰ بار اشکال با توزیع یکنواخت تزریق می‌شود.

۴-۲- محیط آزمایشگاهی تزریق اشکال و تحلیل آن

به منظور دستیابی به نتایج دقیق و بدون وابستگی به نوع سیستم، تلاش بر انتخاب برنامه‌های محک از دسته‌های مختلف شده است تا رفتار آن‌ها در برابر خطاهای گذرا ارزیابی شوند. از طرفی پایتون بعنوان زبانی که قابلیت گسترش و استفاده در اغلب سیستم عامل ها را دارد انتخاب گردیده است تا این ابزار بتواند به سیستم عامل-های دیگر گسترش یابد. پیکربندی محیط اجرایی مورد استفاده در جدول ۱ خلاصه شده است.در نهایت برای ارزیابی نتایج مجموعه برنامه محک MiBench [۵۵] انتخاب و تحت تزریق اشکال قرار گرفته است. جزئیات برنامه محک مورد استفاده در جدول ۲ قابل مشاهده است.

جدول ۱: محیط اجرایی.

محیط برنامه	
سخت‌افزار	CPU i7 Intel 1.7–2.9 GHz RAM 6 G
سیستم‌عامل	Ubuntu 16.1 Linux 64 bit
نرم‌افزار	ابزار تزریق اشکال پیاده‌سازی شده با زبان پایتون
پایگاه داده	My sql برای جمع آوری جزئیات عملگرها و عملوند ها و کد هگزادسیمال آنها

جدول ۲: برنامه محک مورد استفاده در این پژوهش.

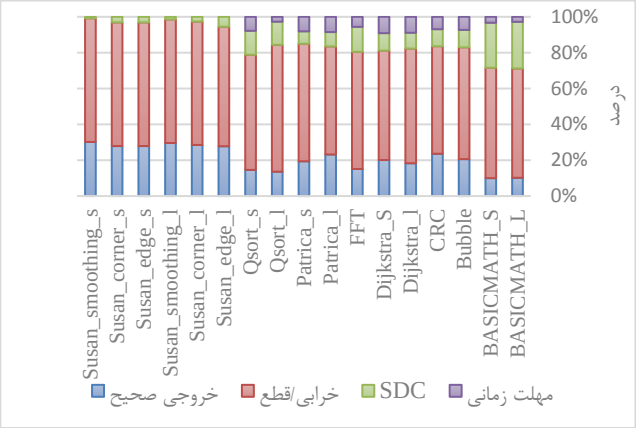
MiBench			other
Auto./Industrial	**Network**	**Telecomm**	
Basicmath(L)	Dijkstra (L)	CRC32	Bubble Sort
Basicmath(S)	Dijkstra (S)	FFT(S)	
Bitcount(L)	Patricia (L)		
Bitcount(S)	Patricia (S)		
Qsort(L)			
Qsort(S)			
Susan smooth, edge,corner(L)			
Susan smooth ,edge,corner(S)			
large-input=L,small-input=S			

۵- نتایج آزمایشگاهی تزریق اشکال

نتایج اشکالات تزریق شده به دو دسته خطای جریان کنترل و خطای داده تقسیم شدند. همانطور که پیشتر گفته شد، خطای جریان کنترل خود منجر به حذف دستورالعمل‌های پرش، ایجاد دستورالعمل‌های جدید پرش یا تغییر مقصد دستورالعمل پرش می‌شوند. بقیه اشکالات نیز جزو خطای داده محاسبه شده است. این دو دسته عمومی به دو زیر گروه از خطاهای یک بیتی و چند بیتی تقسیم شده‌اند.

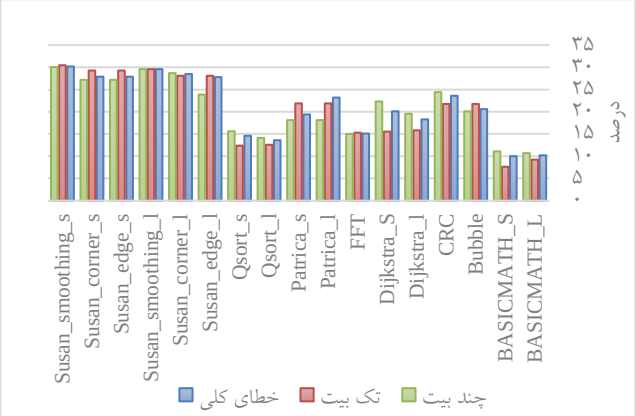
۵-۱- نتایج کلی تزریق اشکال

نتایج کلی حاصل بدون هیچ تقسیم‌بندی، در شکل ۲ نشان داده شده است. با توجه به این نتایج می‌توان گفت که در تمام برنامه-های محک، رفتار تقریباً یکسانی در برابر خطاها دیده می‌شود ، بیشترین سهم مطابق شکل به ترتیب شامل خرابی/ قطع، خروجی صحیح، SDC و در نهایت مهلت‌زمانی است، به استثنای دو برنامه patricia و basicmath که ترتیب بین خروجی صحیح و SDC تغییر کرده است. علت آن است که طبق [۵۵] تعداد عدد صحیح بیشتری در کد منبع وجود دارد. از اینرو حجم داده بیشتر، امکان رخداد SDC را افزایش می‌دهد. از سوی دیگر basicmath دارای محاسبات ریاضی زیادی است که این خود باعث افزایش SDC می‌شود. patricia هم در مقایسه با سایر برنامه‌های محک دارای پیچیدگی کد بیشتری است.



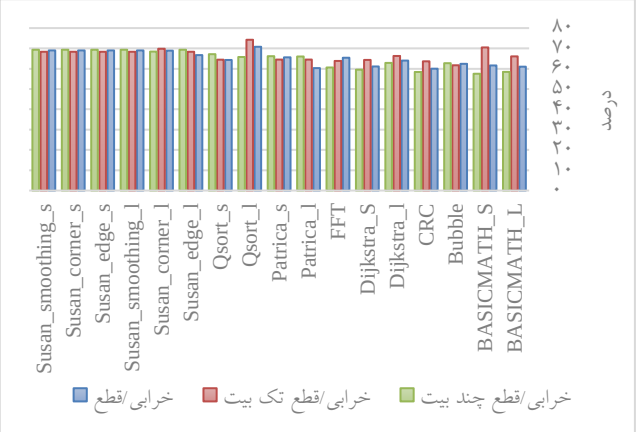
شکل ۲: نتایج کلی تزریق اشکال.

رفتار سیستم در برابر خطاهای چندبیتی و تک‌بیتی در شکل ۳ تا ۶ نشان داده شده است. همانطور که در شکل ۳ نشان داده شده است، ۱۰ تا ۳۰ درصد خطاها معمولاً منجر به هیچ اتفاق خاصی در سیستم نمی‌شوند و خروجی صحیحی را تولید می‌کنند که به آن خروجی خوش‌خیم یا صحیح می‌گویند. به تفکیک سهم این خروجی در تزریق اشکال تک بیتی ۲ تا ۷ درصد و تقریباً یک رفتار مشابه در اشکال چند بیتی که ۲ تا ۱۰ درصد می‌باشد.



شکل ۳: درصد نتایج خروجی صحیح با توجه به تزریق اشکال یک یا چند بیتی.

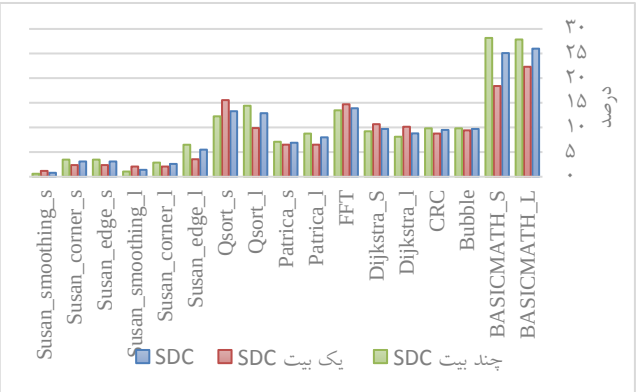
در شکل ۴ مشاهده می‌شود که ۶۰ تا ۷۰ درصد خطاها منجر به خرابی/ قطع سیستم که سهم اشکال تک بیتی ۶۲ تا ۷۸ و چند بیتی ۵۸ تا ۶۹ درصد است. این خطا رایج‌ترین خطا است زیرا مربوط به قسمت اصلی برنامه است که امکان کامپایل برنامه پس از تزریق اشکال وجود ندارد، بنابراین منجر به نتیجه خرابی/ قطع می‌شود.



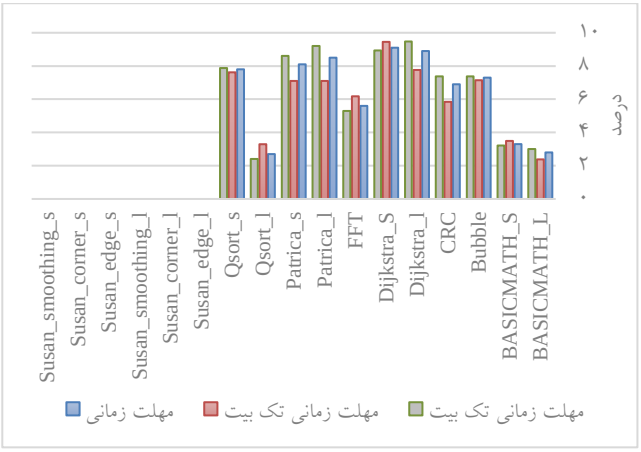
شکل ۴: درصد نتایج خرابی/ قطع با توجه به تزریق اشکال یک یا چند بیتی.

بین ۰.۸ تا ۲۶ درصد خطاها منجر به SDC می‌شوند که تقریباً مشابه رفتار تزریق تک بیتی با ۱ تا ۲۲ درصد و در چند بیتی ۰.۶ تا ۲۸ درصد نتایج SDC است که در شکل ۵ مشاهده می‌شود. ۰ تا ۹ درصد خطاها منجر به خطای نقض زمانی می‌شود که سهم نتایج یک بیتی و چند بیتی یکسان و بین ۴ تا ۹ درصد است و می توان در شکل ۶ مشاهده کرد.

همانطور که قبلا هم گفته شد، تاکنون خطاهای تک بیتی مورد توجه محققان بوده است. بعبارت دیگر، خطاهای چند بیتی کمتر مورد بررسی قرار گرفته‌اند. از اینرو باید رفتار سیستم‌ها در برابر چنین خطاهایی به دلیل افزایش تعداد خطاهای چند بیتی در محیط مورد بررسی قرار داده شوند [۹-۶]. برخلاف ادعای مطرح شده محققان، که خطاهای چند بیتی را به دلیل ماهیت وقوع آن-ها می‌توان نادیده گرفت [۶]. مشاهده شد که خطاهای چند بیتی از لحاظ نرخ رخداد در رفتاری مشابه با خطاهای تک بیتی می‌توانند منجر به خرابی/ قطع، خروجی صحیح و SDC شوند. بنابراین، نادیده
گرفتن آن‌ها منجر به نتایج جبران ناپذیری می‌گردد.



شکل ۵: درصد نتایج SDC با توجه به تزریق اشکال یک یا چند بیتی.



شکل ۶: درصد نتایج نقض زمانی با توجه به تزریق اشکال یک یا چند بیتی.

۵-۲- خطاهای جریان کنترل

مبتنی بر نتایج موجود در جدول ۳ مشاهده می‌شود که تنها ۱۲ تا ۲۱ درصد از خطاهای تزریق شده منجر به خطاهای جریان کنترل می‌شود که سهم کوچکی از نتایج را به خود اختصاص داده است. به عبارت دیگر، اغلب خطاهای تزریق شده منجر به خطاهای داده می‌شوند. به طور میانگین ۱۶٪ از خطاها منجر به خطای جریان کنترل می‌شوند، زیرا مطابق با پیوست ۱، از تعداد کل دستورالعمل‌های یک برنامه فقط بخش کوچکی از آن‌ها باعث خطای جریان کنترل می‌شود. بنابراین، قابل انتظار است که درصد خطاهای کنترل جریان کمتر باشد.

همانطور که در جدول ۴ نشان داده شده، بیشتر خطای جریان کنترل منجر به خرابی/ قطع می‌شوند که به دلیل تغییر در جریان کنترلی برنامه است.

جدول۳: تقسیم بندی اشکالات بر اساس نوع خطا.

دسته بندی خطا	کمینه/ درصد	بیشینه /درصد
خطای جریان کنترل	۱۲	۲۱
خطای داده	۷۹	۸۸

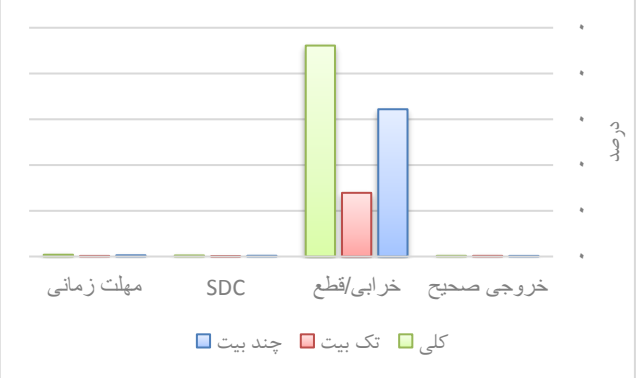
جدول۴: درصد/نتایج خطاهای جریان کنترل.

دسته بندی خطا	کمینه/ درصد	بیشینه /درصد
نتایج درست	۱۰	۴۳
خرابی/قطع	۳۹	۷۶
SDC	۰	۵۰
نقض زمانی	۰	۱۶

دسته‌بندی دقیق‌تر برای خطاهای جریان کنترل شامل حذف پرش، درج پرش و تغییر مقصد پرش است. لذا بر این اساس در ادامه به بررسی تفضیلی این دسته‌بندی پرداخته شده است.

۵-۲-۱- خطاهای درج پرش

همانطور که در شکل ۸ نشان داده شده است، در این دسته از خطاها، نتایج اکثر رویدادها، خرابی/ قطع خواهد بود. زیرا درج یک پرش جدید به طور غیر منتظره جریان برنامه را تغییر می‌دهد. به طوری که برنامه قبل از این هیچ نقشه‌ای برای آن نداشته است. بعبارت دقیق‌تر زمانیکه برنامه به یک دستور پرش برخورد می‌کند برای هر دو حالت درست یا نادرست شرط پرش مسیری را درنظر دارد در نتیجه حذف یا تغییر مقصد پرش معمولا رخداد غیر منتظره‌ای ایجاد نخواهد کرد. اما در صورت درج پرش مسیر جدیدی در برنامه ایجاد می‌گردد. لذا درصد بالایی منجر به خرابی سیستم (خرابی/ قطع) می‌شود.



شکل۸ : نتایج خطای درج پرش.

۵-۲-۲- خطاهای حذف پرش

مطابق جدول ۵، احتمال حذف پرش بیشتر از بقیه است. زیرا در مجموعه دستورالعمل‌ها، احتمال تبدیل یک دستورالعمل پرش به یک دستور غیر پرش، بسیار بیشتر از تبدیل دستورالعمل غیر پرش به دستور پرش است. از طرف دیگر، مطابق شکل ۹ احتمال بدست آوردن خروجی صحیح و خرابی/ قطع تقریباً برابر است. اما یک مسئله جالب احتمال وقوع SDC در این نوع خطا است که بیشتر از درج پرش و تغییر مقصد پرش (حدود ۱۲٪) است که به ترتیب ۰.۱٪ و ۱٪ است. قبلاً توضیح دادیم که دلیل این امر می‌تواند پیش بینی قبلی برنامه در حذف پرش باشد، برای مثال شرط تکمیل حلقه در برنامه. بنابراین، روند عادی برنامه در حضور اشکال منجر به تولید SDC می‌شود.

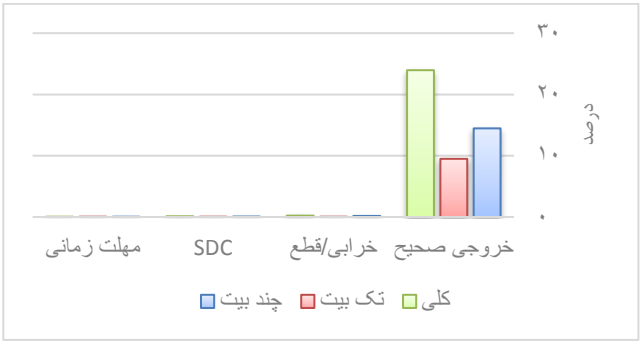
جدول ۵ : درصد نتایج خطای داده.

نتایج به در صد	تک بیتی	چند بیتی
خروجی صحیح	۱۷.۶	۲۱
خرابی/ قطع	۷۱.۲	۶۶
SDC	۷.۳	۹
نقض زمانی	۳.۷	۴

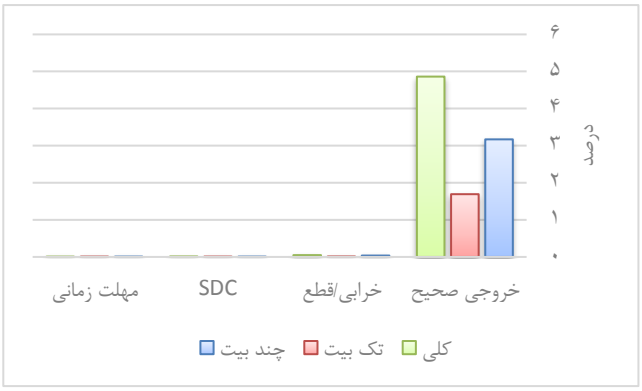
۵-۲-۳- خطاهای تغییر مقصد پرش

احتمال این خطا از بقیه دسته‌های خطا کمتر است؛ زیرا احتمال تبدیل دستور پرش به یک دستورالعمل پرش دیگر طبق پیوست ۱ کم است. همانطور که در بالا گفته شد احتمال خرابی و خروجی صحیح همانطور که در شکل ۱۰ قابل مشاهده‌است، برابر است.

بر اساس شکل‌های ۸ تا۱۰ نرخ خرابی که خطای تک بیت تحمیل می‌کند کمتر از خطای چند بیتی است زیرا احتمال تغییر یک دستورالعمل پرش به یک دستور غیر پرشی یا دستورالعمل پرشی دیگر بیشتر از تغییر دستورالعمل غیر پرشی به یک دستور پرشی است. نکته جالب دیگر درصد بالای SDC در خطای چند بیتی مانند تک بیتی است که نشان می‌دهد احتمال تولید SDC در خطاهای چند بیتی نیز زیاد و غیر قابل اغماض است.



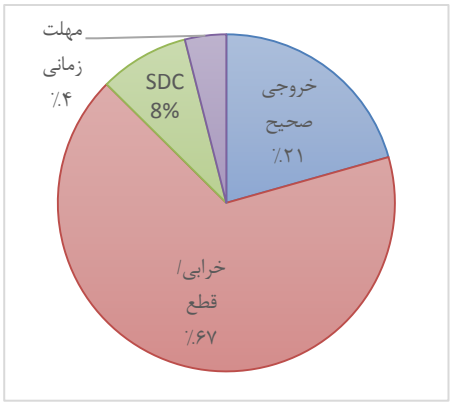
شکل۹ : نتایج خطای حذف پرش.



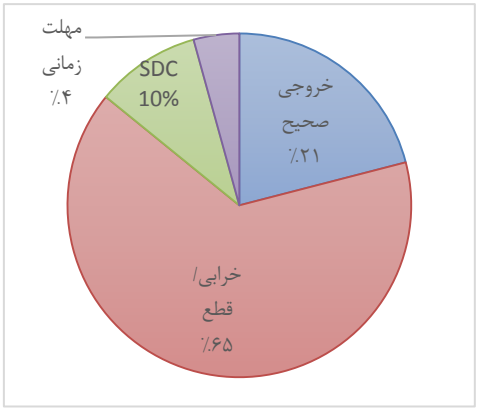
شکل ۱۰ : نتایج خطای تغییر مقصد پرش.

۵-۳- خطاهای داده

مطابق جدول ۳ ، ۷۹٪ تا ۸۸٪ از خطاهای تزریق شده منجر به خطاهای داده می‌گردند. با توجه به نتایج برخی تحقیقات، همانگونه که انتظار می‌رفت طبق نتایج جدول ۵، درصد خطاهای خرابی/ قطع کمتر از نتایج جدول ۵ خروجی صحیح و SDC در حالت چند بیتی بیشتر از تک بیت هستند.



الف



ب

شکل ۱۱: نمودار دایره ای الف خطاهای تک بیتی و نمودار دایره ای ب خطاهای چند بیتی

همانطور که در دو نمودار دایره‌ای شکل ۱۱ قابل مشاهده است، نرخ رخداد خرابی درخطاهای چند بیتی کمتر از خطاهای تک بیتی است. از سوی دیگر نرخ رخداد خروجی صحیح برابر است. نکته قابل تامل در این نمودارها نرخ رخداد SDC است. نرخ رخداد SDC در حالت چند بیتی ۲ درصد بیشتر از تک بیت است و باید توجه کرد که در مجموع کل خطاها این رخداد افزایش پیدا می‌کند. لذا به دلایل مسایلی که رخداد خروجی نادرست در سیستم‌های کامپیوتری ایجاد می‌کند. تشخیص و کاهش رخداد این خطا لازم است. در تزریق اشکالات تک بیتی و چند بیتی برخلاف نتایج برخی تحقیقات، احتمال بروز SDC متفاوت است. که نشان می‌دهد خطاهای چند بیتی نیز باید مورد بررسی قرار گیرند.

۶- جمع‌بندی

در این مقاله رفتار سیستم‌ها در برابر خطاهای چند بیتی بررسی شده است. از آنجا که کارهای گذشته اغلب فقط به خطاهای نرم در بخشی از دستورالعمل ها مانند بارگذاری/ ذخیره کردن و تنها چند مقاله به تأثیرات خطاهای چند بیتی پرداخته است، لذا در این تحقیق علاوه بر در نظر گرفتن خطاهای چند بیتی به تزریق اشکال به کلیه دستورالعمل‌ها نیز پرداخته شده است تا بتوان رفتار سیستم را با دقت بیشتری بررسی نمود. برای این منظور، با استفاده از طراحی ابزار تزریق در پایتون و ایجاد یک پایگاه داده MySQL ، اشکالات به دستورالعمل‌ها و داده‌های، برنامه‌های محک MiBench تزریق شده است. از نتایج به دست آمده می‌توان دریافت که رفتار سیستم در برابر خطاهای چند بیتی مشابه خطاهای تک بیتی است. اما به گونه‌ای نیست که بتوان آنها را نادیده گرفت. به طوری میانگین میزان SDC، در خطاهای چند بیتی ۲ درصد افزایش یافته است. افزایش میزان وقوع این نتیجه، تأثیر مستقیم بر روی قابلیت اطمینان دارند. همچنین، مشاهده شد که میزان خطای نرم در اشکالات چند بیتی افزایش می‌یابد. لذا می‌بایست فکری برای اثرات این خطاها نمود. در نهایت پیشنهاد می‌شود، در کارهای پیش رو بر روی روش‌هایی برای هرس محل تزریق و ارائه روشی برای پیش بینی نتایج خطاهای چند بیتی تمر کز شود.

مراجع

[1] A. Dixit and A. Wood, “The impact of new technology on soft error rates,” in Proc. IEEE Int. Rel. Phys. Symp. (IRPS), pp. 5B.4.1–5B.4.7, 2011.

[2] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, “Near threshold voltage (NTV) design: Opportunities andChallenges,” in Proc. 49th Annu. Design Autom. Conf. (DAC), pp. 1153–1158, 2012.

[3] N. Aggarwal and P. Ranganathan and N. P. Jouppi and J. E. Smith, “Configurable Isolation: Building High Availability Systems withCommodity Multi Core Processors”, 34th Annual InternationalSymposium on Computer Architecture, pp.340–347, 2007.

[4] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, “Impact of scaling on neutron induced soft error in SRAMs from a 250 nm to a 22 nm design rule,” IEEE Trans. Electron Devices, vol. 57, no. 7, pp. 1527–1538, July, 2010.

[5] D. Radaelli, H. Puchner, S. Wong, and S. Daniel, “Investigation of multi¬bit upsets in a 150 nm technology SRAM device,” IEEE Trans.Nucl. Sci., vol. 52, no. 6, pp. 2433–2437, Dec. 2005.

[6] B. Sangchoolie, K. Pattabiraman and J. Karlsson, “One Bit is (Not) Enough: An Empirical Study of the Impact of Single and Multiple Bit Flip Errors”, 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), USA, pp.26–29, 2017.

[7] H. Cho , S. Mirkhani, C.Y. Cher, J. A. Abraham and S. Mitra, “Quantitative evaluation of soft error injection techniques for robust

[26] S.A. Asghari, H.R.Zarandi, H.Pedram, M.Ansarinia and M.Khademi, "A Fault Injection Attitude based on Background Debug Mode in Embedded Systems", Proceedings of the International Conference on Computer Design, DES, USA, 2009.

[27] S. A. Asghari, A. Abdi, H. Taheri, H. Pedram, S. Pourmozaffari, "SEDSR: Soft Error Detection Using Software Redundancy", Journal of Software Engineering and Applications, Vol.5, pp.664–670,2012.

[28] Sh. Feng, Sh. Gupta, A. Ansari and S. Mahlke, "Shoestring: Probabilistic soft error reliability on the cheap", In Proceedings of the Fifteenth Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems ASPLOS XV,pp.305–336,2010.

[29] S. K. SastryHari, S. V. Adve, H. Naeimi and P. Ramachandra,Relyzer: Exploiting application¬level fault equivalence to analyze application resiliency to transient faults", In Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems ASPLOS XVII ,pp.123–134,2012.

[30] M. Maghsoudloo, H. R. Zarandi, S. T. Pour Mozafari and N. Khoshavi, "Soft Error Detection Technique in Multi threaded Architectures Using Control Flow Monitoring", 14th Euromicro Conference on Digital System Design, pp.789–792, 2011.

[31] M. Maghsoudloo, H.R. Zarandi and N. Khoshavi,"Low Cost Software Implemented Error Detection Technique", International Symposium on Electronic System Design,pp.318–323,2011.

[32] J. Vankeirsbilck, N.Penneman, H.Hallez and J.Boydens, "Random Additive Signature Monitoring for Control Flow Error Detection", TRANSACTIONS ON RELIABILITY, Vol.66, No.4, pp=1178–1192, 2017.

[33] Q. Lu, G. Li, K.Pattabiraman, M. S. Gupta and .J. A. Rivers,"SDCTune: A model for predicting the SDC proneness of an application for configurable protection", International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES), India2014.

[34] A. Thomas and K. Pattabiraman, "Error detector placement for soft computation, 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Hungary, 2013.

[35] N.Narayanamurthy, K. Pattabiraman and M.Ripeanu, "Finding Resilience¬Friendly Compiler Optimizations Using Meta–Heuristic Search Techniques", 12th European Dependable Computing Conference, Sweden, 2016.

[36] I. Laguna, I. Laguna and D.F. Richards ,"IPAS: Intelligent protection against silent output corruption in scientific applications, IEEE/ACM International Symposium on Code Generation and Optimization (CGO),Spain, 2016.

[37] N. Seifert et al., "Soft error susceptibilities of 22 nm trigate devices," IEEE Trans. Nucl. Sci., Vol. 59, No. 6, pp. 2666–2673, Dec. 2012.

[38] Lu, Q., Li, G., Pattabiraman, K., Gupta, M. S., and Rivers, J. A., (2017), "Configurable Detection of SDC-causing Errors in Programs", ACM Transactions on Embedded Computing Systems, Vol. 9, No. 4, Article 39, March.

[39] GU, J., Zheng, W., Zhuang, Y., Zhang, Q., (2019),"Vulnerability analysis of instructions for SDC-causing error detection", IEEE Access, Volume 7.

[40] Liu, L., Ci, L., Liu, W., Yang, H., 2019,"Identifying SDC-causing Instructions based on Random forests algorithm", KSII Transactions on Internet and Information Systems. Vol. 13.

[41] Yang, N., Wang, Y., (2019),"Identify Silent Data Corruption Vulnerable Instructions Using SVM", IEEE Access, Volume 7.

[42] Wang, C., Dryden, N., Cappello, F., Snir, M., (2018), "Neural network based silent error detector", IEEE International Conference on Cluster Computing (CLUSTER.(

system design", in Proceedings of the 50th ACM/EDAC/IEEE Design Automation Conference, pp.1–10, 2013.

[8] G. A. Kanawati, N. A. Kanawati and J. A. Abraham, "EMAX: An automatic extractor of high level error models", in Proceedings of the 9th AIAA Computing in Aerospace Conference, pp.1297–1306, 1993.

[9] J. F. Ziegler et al,"IBM experiments in soft fails in computer electronics (1978–1994)", IBM Journal of Research and Development, Vol.40, No.1, pp.3–18, 1996.

[10] A. Chatzidimitriou, G. Papadimitriou, C. Gavanas, G. Katsoridas and D. Gizopoulos, "Multi-Bit Upsets Vulnerability Analysis of Modern Microprocessors," 2019 IEEE International Symposium on Workload Characterization (IISWC), pp. 119-130,,2019.

[11] Chabot, A., Alouani, I., Nouacer, R. et al. A Memory Reliability Enhancement Technique for Multi Bit Upsets. J Sign Process Syst 93,pp. 439–459 (2021).

[12] S. A. Asghari, H. Taheri, H.Pedram and O. Kaynak,"Software-Based Control Flow Checking Against Transient Faults in Industrial Environments",IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, Vol.10,No.1,pp.481–490,2014.

[13] Q. Lu, G. Li and K. Pattabiraman, M. S. Gupta and .J. A. Rivers ,"Configurable Detection of SDC causing Errors in Programs" , ACM Transactions on Embedded Computing Systems (TECS), Vol.16,2017.

[14] S. S. Mukherjee, J. Emer and S. K. Reinhardt,"The Soft Error Problem: An Architectural Perspective", the 11th International Symposium on High¬Performance Computer Architecture (HPCA), USA, 2005.

[15] A.Rohani, "Modelling and Mitigation of Soft Errors in CMOS Processors", Ph.D. Thesis ,University of Twente, 2014.

[16] X. Meng, Q. Tan, Z. Shao, N. Zhang, J. Xu and H. Zhang, "SEInjector: A Dynamic Fault Injection Tool for Soft Errors on X86", 2017 International Conference on Computer Systems, Electronics and Control (ICCSEC),India,pp.1492–1495,2017.

[17] X. Meng, Q. Tan, Z. Shao, N. Zhang, J. Xu, H. Zhang,"Optimization Methods for the Fault Injection Tool SEInjector", 2018 International Conference on Information and Computer Technologies (ICICT), USA, pp.31–35,2018.

[18] Chun Kai Chang, Sangkug Lym, Nicholas Kelly, Michael B. Sullivan1 and Mattan Erez, "Hamartia: A Fast and Accurate Error Injection Framework", 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, Luxembourg, pp.101–108, 2018.

[19] J. Aidemark, J. Vinter, P. Folkesson, J. Karlsson, "GOOFI: generic object–oriented fault injection tool", International Conference on Dependable Systems and Networks, 2001.

[20] D. Skarin, R. Barbosa and lohan Karlsson,"GOOFI–2: A Tool for Experimental Dependability Assessment, IEEEIIFIP International Conference on Dependable Systems & Networks DSN,USA, pp557–562,2010.

[21] F. Ayatolahi, B. Sangchoolie, R. Johansson and J. Karlsson, "A Study of the Impact of Single Bit–Flip and Double Bit¬Flip Errors on Program Execution", SAFECOMP, pp.265–276, 2013.

[22] F.Adamu Fika and Arshad Jhumka, "An Investigation of the Impact of Double Single Bit–Flip Errors on Program Executions", DEPEND: The Eighth International Conference on Dependability, 2015.

[23] B. Sangchoolie, F. Ayatolah, R. Johansson; Johan Karlsson,"A Study of the Impact of Bit¬Flip Errors on Programs Compiled with Different Optimization Levels", Tenth European Dependable Computing Conference, 2014.

[24] B. Sangchoolie, K. Pattabiraman and J. Karlsson, "An Empirical Study of the Impact of Single and Multiple Bit-Flip Errors in Programs," in *IEEE* Transactions on Dependable and Secure Computing (TDSC), 2020

[25] S.A. Asghari and H.Taheri,"An Effective Soft Error Detection Mechanism using Redundant Instructions", International Arab Journal of Information Technology, Vol.12, No.1, 2015.

[43]Thomas, A. and Pattabiraman, K., (2013), "Error Detector Placement for Soft Computation", Dependable Systems and Networks (DSN), 43rd Annual IEEE/IFIP International Conference on, June 2013

[44] Thati, V.B., Vankeirsbilck, J., Boydens, J., Pissort, D., (2019),"Selective duplication and selective comparison for data flow", 2019 4th International Conference on System Reliability and Safety (ICSRS.(

[45] Ma, J., Duan, Z., Tang, L., (2019),"Amethodology to assess output vulnerability factors for detecting silent data corruption", IEEE Access, Volume 7.

[46] Li, G., Pattabiraman, K., Hari, S.K.S., Sullivan, M., Tsai, T., (2018),"Modeling Soft-error Propagation in programs", 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).

[47] M. Ebrahimi, H. Asadi, and M. B. Tahoori, "A layout–based approach for multiple event transient analysis," in Proc. 50th Annu. Design Autom. Conf. (DAC), pp. 1–6, 2013.

[48] N. Khoshavi and A. Samiei, "The Study of Transient Faults Propagation in Multithread Applications", arXiv, 2016.

[49] B. Fang, K.Pattabiraman, M.Ripeanu, and S. Gurumurthi, "GPU–Qin A Methodology for Evaluating the Error Resilience of GPGPU Applications", IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), June 2014.

[50] G.A. Reis, J. Chang, N. Vachharajani, R. Rangan and D.I. August,"SWIFT: software implemented fault tolerance", International Symposium on Code Generation and Optimization, 2005.

[51] A. Thomas, K. Pattabirama, "LLFI: An intermediate code level fault injector for soft computing applications", Workshop on Silicon Errors in Logic System Effects (SELSE), 2013.

[52] A. Banaiyan Mofrad,M., Ebrahimiy, F., Oborily, M.B., Tahooriy, and N., Dutt, "Protecting Caches Against Multi¬Bit Errors Using Embedded Erasure Coding", 20th IEEE European Test Symposium (ETS),2015.

[53] M. Ebrahimi, P. Murali, B. Rao, R.Seyyedi and M. B. Tahoori,"Low–Cost Multiple Bit Upset Correction in SRAM¬Based FPGA Configuration Frames".IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.24 , No. 3 ,pp.932–943,2016.

[54] MazeGen, 2017,Fev,http://www.ref.x86asm.net/.

[55] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austi, T. Mudge and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite", Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization, pp.3–14,2001.

پیوست ۱

در این پیوست احتمال تبدیل کد باینری یک دستور با تزریق اشکال یک و چند بیتی به کد یک دستور معتبر دیگر محاسبه شده است.

احتمال تبدیل نک بیتی	احتمال تبدیل چند بیتی	کد باینری	دستورالعمل
0.875	0.857143	86	XCHG
0.875	0.857143	87	XCHG
100	0.904762	88	MOV
100	0.857143	89	MOV
100	0.857143	8A	MOV
100	0.857143	8B	MOV
۱۰۰	۰،۸۵۷۱۴۳	C۸	MOV

احتمال تبدیل نک بیتی	احتمال تبدیل چند بیتی	کد باینری	دستورالعمل
100	0.857143	8D	LEA
100	0.857143	8E	MOV
100	0.857143	8F	POP
0.5	0.619048	90	NOP
0.875	0.857143	98	CBW
0.75	0.809524	99	CWD
0.75	0.809524	9B	FWAIT
0.75	0.857143	9C	PUSHF
0.875	0.809524	A4	MOVS
0.875	0.809524	A5	MOVS
0.75	0.761905	A6	CMPS
0.875	0.809524	A7	CMPS
0.875	0.904762	A8	TEST
0.875	0.904762	A9	TEST
0.875	0.904762	AA	STOS
0.875	0.904762	AB	STOS
0.875	0.904762	AC	LODS
0.875	0.904762	AD	LODS
0.75	0.857143	AE	SCAS
0.875	0.904762	AF	SCAS
0.875	0.904762	C0	ROL
0.875	0.952381	C1	ROL
0.875	0.904762	C2	RETN
0.875	0.904762	C3	RETN
0.875	0.952381	C6	MOV
0.875	0.952381	C7	MOV
0.875	0.952381	C8	ENTER
0.875	0.952381	C9	LEAVE
0.875	0.952381	CA	RETF
0.875	0.952381	CB	RETF
0.875	0.952381	CC	INT
0.875	0.952381	CD	INT
0.875	0.952381	CE	INTO
0.875	0.952381	CF	IRET
0.75	0.857143	D0	ROL
0.75	0.857143	D1	ROL
0.625	0.809524	D2	ROL
0.625	0.809524	D3	ROL
0.75	0.857143	D7	XLAT
0.875	0.904762	D8	FADD
0.875	0.904762	D9	FLD
0.875	0.904762	DA	FIADD
0.875	0.904762	DB	FILD

دستورالعمل	کد باینری	احتمال تبدیل چند بیتی	احتمال تبدیل نک بیتی
CLD	FC	0.904762	0.875
STD	FD	0.952381	0.875
INC	FE	0.857143	0.875
INC	FF	0.904762	0.875
POPF	9D	0.809524	0.75
SAHF	9E	0.809524	0.75
LAHF	9F	0.809524	0.75
MOV	A0	0.904762	0.875
MOV	A1	0.857143	0.875
MOV	A2	0.857143	0.875
MOV	A3	0.857143	0.875
XOR	35	0.857143	0.875
CMP	38	0.857143	0.875
CMP	39	0.904762	0.875
CMP	3A	0.809524	0.75
CMP	3B	0.904762	0.875
CMP	3C	0.857143	0.75
CMP	3D	0.857143	0.875
MOVSXD	63	0.809524	0.75
PUSH	68	0.857143	0.875
IMUL	69	0.857143	0.875
PUSH	6A	0.857143	0.875
IMUL	6B	0.857143	0.875
INS	6C	0.857143	0.75
INS	6D	0.857143	0.75
OUTS	6E	0.809524	0.625
OUTS	6F	0.857143	0.75
JO	70	0.809524	0.75
JNO	71	0.857143	0.875
JB	72	0.809524	0.75
JNB	73	0.809524	0.75
JBE	76	0.761905	0.625
JNBE	77	0.809524	0.75
JS	78	0.857143	0.875
JNS	79	0.857143	0.875
JP	7A	0.857143	0.875
JNP	7B	0.857143	0.875
JL	7C	0.809524	0.875
JNL	7D	0.809524	0.875
JLE	7E	0.761905	0.75
JNLE	7F	0.809524	0.875
ADD	80	0.904762	100
ADD	81	0.857143	0.875

دستورالعمل	کد باینری	احتمال تبدیل چند بیتی	احتمال تبدیل نک بیتی
FADD	DC	0.904762	0.875
FLD	DD	0.904762	0.875
FIADD	DE	0.904762	0.875
FILD	DF	0.904762	0.875
LOOPNZ	E0	0.952381	0.875
LOOPZ	E1	100	100
LOOP	E2	0.952381	0.875
JECXZ	E3	0.952381	0.875
IN	E4	0.952381	0.875
IN	E5	0.952381	0.875
OUT	E6	0.904762	0.875
OUT	E7	0.952381	0.875
CALL	E8	0.952381	100
JMP	E9	100	100
JMP	EB	0.952381	100
IN	EC	100	100
IN	ED	100	100
OUT	EE	0.952381	100
OUT	EF	100	100
undefined	F1	0.761905	0.625
HLT	F4	0.809524	0.75
CMC	F5	0.857143	0.875
TEST	F6	0.761905	0.75
TEST	F7	0.857143	0.75
CLC	F8	0.904762	0.75
STC	F9	0.952381	0.875
CLI	FA	0.904762	0.75
STI	FB	0.904762	0.75
CLD	FC	0.904762	0.875
STD	FD	0.952381	0.875
INC	FE	0.857143	0.875
INC	FF	0.904762	0.875
undefined	F1	0.761905	0.625
HLT	F4	0.809524	0.75
CMC	F5	0.857143	0.875
TEST	F6	0.761905	0.75
TEST	F7	0.857143	0.75
CLC	F8	0.904762	0.75
STC	F9	0.952381	0.875
STC	F9	0.952381	0.875
CLI	FA	0.904762	0.75
STI	FB	0.904762	0.75

دستورالعمل	کد باینری	احتمال تبدیل چند بیتی	احتمال تبدیل نک بیتی
ADD	83	0.857143	0.875
TEST	84	0.857143	0.875
TEST	85	0.857143	0.875
JNZ	75	0.809524	0.75
JZ	74	0.809524	0.75

پاورقی‌ها:

[1] Soft Error Rate
[2] Control Flow Error
[3] Benign / Mask
[4] Crash / Hang
[5] Silent Data Corruption
[6] Timeout
[7] Python
[8] Branch Insertion
[9] Branch Deletion
[10] Branch Target Modification
[11] Simulation
[12] Emulation
[13] Register Transfer Level
[14] Double Bit Single Error
[15] Triple Module Redaundancy
[16] Pin Framework
[17] Gait