دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر
ساختارو زبان کامپیوتر دکتر اسدی

عنوان پروژه امتیازی:

پیاده سازی ساده MIPS

اعضای گروه

نادر احمدی ۴۰۳۱۰۵۶۷۴

محمدپارسا محمودآبادی آرانی ۴۰۳۱۰۶۶۷۹

سیدامیرحسین هاشمی شاهرودی ۴۰۳۱۷۰۲۷۱

پارسا پاک ۴۰۳۱۰۵۸۲۲

بهمن ۱۴۰۴

چکیده

در این پروژه، یک شبیه‌ساز نرم‌افزاری برای زیرمجموعه‌ای از دستورات معماری MIPS32 طراحی و پیاده‌سازی شده است. هدف اصلی، مدل‌سازی دقیق اجزای سخت‌افزاری پردازنده شامل حافظه اصلی (RAM)، مجموعه ثبات‌ها (Register File) و شمارنده برنامه (PC) در یک محیط نرم‌افزاری است. این شبیه‌ساز قادر است دستورات ماشین را در قالب هگزادسیمال دریافت کرده و با طی کردن چرخه کامل Fetch-Decode-Execute، تغییرات متناظر را در وضعیت ثبات‌ها و حافظه اعمال کند.

۱ مقدمه Introduction

هدف اصلی این پروژه، ابهام‌زدایی از نحوه عملکرد پردازنده در بنیادی‌ترین سطح است. در این راستا، یک شبیه‌ساز نرم‌افزاری با زبان برنامه‌نویسی پایتون طراحی شده است که قادر است دستورات ماشین (Machine Instructions) را از حافظه خوانده، آن‌ها را رمزگشایی (Decode) کرده و عملیات متناظر را اجرا نماید. این شبیه‌ساز زیرمجموعه‌ای از معماری MIPS32 را پشتیبانی می‌کند.

۲ اجزای سیستم System Components

- **حافظه اصلی (Main Memory):** آرایه‌ای ۱۰۲۴ عضوی از نوع ۳۲ بیتی (۴ کیلوبایت)، به صورت بایت‌آدرس‌پذیر.
- **مجموعه ثبات‌ها (Register File):** شامل ۳۲ ثبات ۳۲ بیتی که ثبات صفر همواره مقدار صفر دارد.
- **شمارنده برنامه (PC):** نگهدارنده آدرس دستور جاری (شروع از صفر).

۳ قالب دستورات Instruction Formats

۱. **نوع R:** عملیات بین ثبات‌ها مانند ADD, SUB, SLT

۲. **نوع I:** شامل مقدار فوری و دسترسی حافظه مانند ADDI, LW, SW, BEQ

۳. **نوع J:** پرش غیرشرطی J

۴ دستورات پیاده‌سازی شده Implemented Instructions

جدول ۱: جدول دستورات پشتیبانی شده

Operation	Funct	Opcode	Type	Instruction
R[rt] + R[rs] = R[rd]	0x20	0x00	R	ADD
R[rt] – R[rs] = R[rd]	0x22	0x00	R	SUB
R[rt]) > (R[rs] = R[rd]	0x2A	0x00	R	SLT
Imm + R[rs] = R[rt]	–	0x08	I	ADDI
Imm] + Mem[R[rs] = R[rt]	–	0x23	I	LW
R[rt] = Imm] + Mem[R[rs]	–	0x2B	I	SW
۲ » Imm =+ PC if(R[rs]==R[rt])	–	0x04	I	BEQ
(۲ » (Addr = PC	–	0x02	J	J

۵ چرخه اجرا Execution Cycle

واکشی (Fetch): خواندن دستور از حافظه با توجه به PC

رمزگشایی (Decode): استخراج فیلدهای مختلف با عملگرهای بیتی

اجرا (Execute): انجام عملیات یا تغییر PC

توقف (Halt): در صورت واکشی دستور صفر

۶ نمونه آزمایش اولیه Initial Test Case

کد نمونه زیر برای بررسی صحت عملکرد دستورات اجرا شد:

20080005 20090003 01095022 0109582A AC0A0000 00000000

توضیح عملیات:

- ADDI $t0, $zero, 5
- ADDI $t1, $zero, 3
- SUB $t2, $t0, $t1
- SLT $t3, $t0, $t1
- SW $t2, 0($zero)

```
❯ python3 ./mips_sim.py ./sample_input.txt
Final register values:
R00 = 0
R01 = 0
R02 = 0
R03 = 0
R04 = 0
R05 = 0
R06 = 0
R07 = 0
R08 = 5
R09 = 3
R10 = 2
R11 = 0
R12 = 0
R13 = 0
R14 = 0
R15 = 0
R16 = 0
R17 = 0
R18 = 0
R19 = 0
R20 = 0
R21 = 0
R22 = 0
R23 = 0
R24 = 0
R25 = 0
R26 = 0
R27 = 0
R28 = 0
R29 = 0
R30 = 0
R31 = 0
```

شکل ۱: خروجی نهایی ثبات‌ها در کنسول

۷ راهنمای اجرا Execution Guide

اجرای خط فرمان

```
python3 mips_sim.py sample_input.txt
```

اجرای رابط گرافیکی

```
python3 mips_gui.py
```

۸ نمونه تکمیلی: مجموع ۱ تا ۳۰۰ Sum 1 to 300

20030000 20010001 2002012D 00611820 20210001 10220001 08000003 00000000

نتیجه نهایی:

- $\mathrm{R3} = 45150$

۹ نتیجه‌گیری Conclusion

پیاده‌سازی این شبیه‌ساز باعث درک بهتر ساختار پردازنده، چرخه اجرا و مدیریت حافظه شد. چالش اصلی پروژه، پیاده‌سازی صحیح پرش‌ها و مدیریت دقیق آدرس‌دهی حافظه بود.