به نام خدا

دانشگاه شیراز
دانشکده ی مهندسی برق و کامپیوتر دانشگاه شیراز
بخش مهندسی کامپیوتر و فناوری اطلاعات

پروژه ی نهایی درس مدار های منطقی

سال تحصیلی 1398-1399

نیم سال اول

استاد درس: جناب آقای دکتر خونجوش

دستیاران : علیرضا خالقی – محمدحسین اله اکبری

فاز اول:

یک شبه پردازنده با معماری Little Endian هشت بیتی را پیاده‌سازی کنید که در هر پالس ساعت یک دستور بیست بیتی را خوانده، محاسبه کند و جواب آخرین دستور را sign extend کرده و بر روی یک خروجی شانزده بیتی قرار دهد و همچنین فلگ های مربوط به هر محاسبه را نیز تنظیم کند.

جزئیات پیاده سازی :

1. دستوراتی که پردازنده باید آن‌ها را اجرا کند در فایلی به نام "instructions.txt" ذخیره شده‌اند و حداکثر تعداد دستورات نوشته شده در فایل نیز 24 دستور میباشد.

2. پردازنده پس از اجرای هر دستور محاسباتی، دو فلگ به نام های ZF و SF را تنظیم میکند. فلگ اول صفر بودن حاصل را نمایش میدهد(در صورت صفر بودن 1 و در غیر اینصورت 0) و فلگ دوم علامت عدد محاسبه شده را نمایش میدهد. (منفی 1و مثبت 0)

3. این پردازنده دارای چهار رجیستر هشت بیتی به نام های Rc, Rb, Ra و Rd میباشد و هر یک از این رجیستر ها یک شناسه ی 8 بیتی دارند که در قسمت های بعدی ذکر شده اند.

4. دستورات پردازنده از سه بخش تشکیل شده اند:

1. کد دستور:
   هر دستور دارای یک کد 4 بیتی است که پردازنده با توجه به آن تصمیم میگرد که چه عملیاتی بر روی عملوند ها انجام دهد. همچنین کد دستور تعیین میکند که عملوند دوم شناسه ی رجیستر است یا یک عدد هشت بیتی است. کد دستور در ابتدای هر دستور می‌آید و بیت 20 ام تا 17 ام را به خود اختصاص میدهد.

2. عملوند اول:
   عملوند اول هر دستور، شناسه یکی از رجیستر های پردازنده است و تعیین میکند که حاصل هر دستور در کجا ذخیره شود. عملوند اول بیت‌های 16 ام تا 9 ام هر دستور را به خود اختصاص میدهد.

3. عملوند دوم:
   عملوند دوم هر دستور، در همه ی دستورات(به جز دستور shift) میتواند شناسه ی یک رجیسترو یا یک عدد 8 بیتی با علامت باشد. عملوند دوم بیت‌های 8 ام تا 1 ام را به خود اختصاص میدهد.

1. شناسه ی رجیستر ها:

نام رجیستر	شناسه ی رجیستر (regId)
Ra	**00000001**
Rb	**00000010**
Rc	**00000100**
Rd	**00001000**

2. لیست دستورات:

دستور	فرمت دستور	عملیات دستور
Load reg, op	0001_regId_operand	ذخیره ی مقدار op در رجیستر
Load reg1, reg2	0010_regId1_regId2	ذخیره ی مقدار رجیستر 2در رجیستر1
Add reg, op	0011_regId_operand	جمع کردن مقدار رجیستر با مقدار op و ذخیره حاصل در رجیستر
Add reg1, reg2	0100_regId1_regId2	جمع کردن مقدار رجیستر 1 با رجیستر 2 و ذخیره ی حاصل در رجیستر 1
Sub reg, op	0101_regId_operand	کم کردن مقدار op از رجیستر و قرار دادن حاصل در رجیستر
Sub reg1, reg2	0110_regId1_regId2	کم کردن مقدار رجیستر 2 از رجیستر 1 و قرار دادن حاصل در رجیستر 1
Mult reg, op	0111_regId_operand	ضرب کردن مقدار رجیستر در مقدار op و ذخیره ی هشت بیت پر ارزش در Rd و ذخیره ی هشت بیت کم ارزش در Ra
Mult reg1, reg2	1000_regId1_regId2	ضرب کردن مقدار رجیستر 1در مقدار رجیستر 2 و ذخیره ی هشت بیت پر ارزش در Rd و ذخیره ی هشت بیت کم ارزش در Ra
Div reg, op	1001_regId_operand	تقسیم صحیح مقدار رجیستر بر مقدار op و ذخیره ی حاصل در رجیستر
Div reg1, reg2	1010_regId1, regId2	تقسیم صحیح مقدار رجیستر 1 به مقدار رجیستر 2 و ذخیره ی حاصل در رجیستر 1
Shr reg, op	1011_regId_operand	شیفت به راست مقدار رجیستر به اندازه مقدار op
Shl reg, op	1100_regId_operand	شیفت به چپ مقدار رجیستر به اندازه مقدار op

مثال ها:

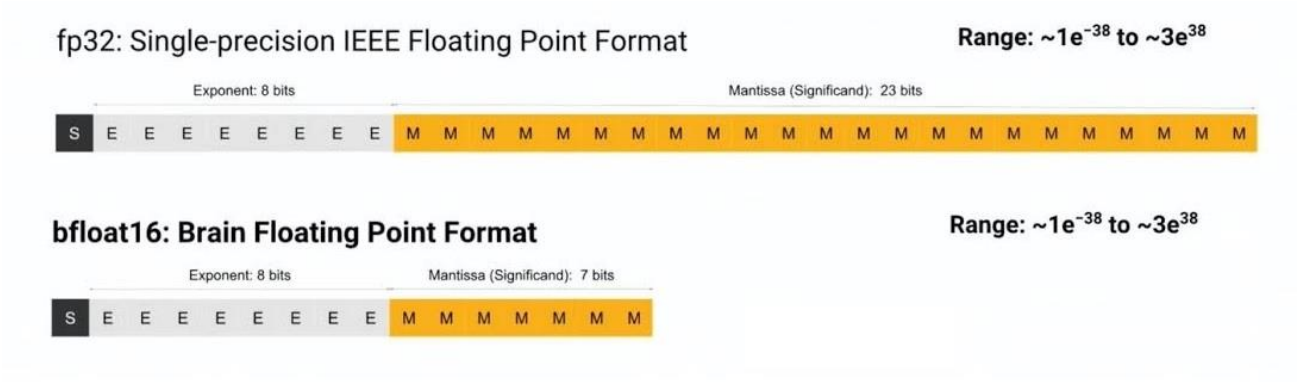
عملیات	دستور	دستور به صورت کد شده
جمع دو عدد 2 و 15	Load Ra, 2 Add Ra, 15	0001_00000001_00000010 0011_00000001_00001111

تفریق 4 از 8 و سپس ضرب آن در 5	Load Rb, 8
	Load Rc, 4
	Sub Rb, Rc
	Mult Rb, 5
		0001_00000010_00001000
		0001_00000100_00000100
		0110_00000010_00000100
		0111_00000010_00000101

لود کردن عدد 7 در رجیستر Ra و شیفت آن به چپ به اندازه ی 3 واحد	Load Ra, 7 Shl Ra, 3	0001_00000001_00000111 1100_00000001_00000011

فاز دوم:

یکی از ویژگی‌های مهم پردازنده های امروزی، انجام محاسبات اعشاری است. اولین گام در انجام محاسبات اعشاری، در نظر گرفتن استانداردی برای نمایش این اعداد است. از استاندارد های معروف میتوان به IEEE 754، که در سه حالت 16 بیتی، 32 بیتی و 64 بیتی قابل استفاده است اشاره کرد. این استاندارد دارای تقریب دقیقی است اما به علت همین تقریب دقیق انجام محاسبات را کند میکند و این امر موجب می‌شود تراشه های مخصوصی که برای اجرای الگوریتم های یادگیری ماشین و شبکه‌های عصبی استفاده می‌شوند در اجرا با کندی مواجه شوند. برای حل این مشکل شرکت گوگل استاندارد جدیدی تحت عنوان bfloat16 را معرفی کرد که مشابه استاندارد IEEE754 single-precision است اما با این تفاوت که به جای 23 بیت، تنها 7 بیت mantissa را نگه داری میکند. به این ترتیب با وجود 16 بیتی بودن، هم بازه ی گسترده ای از اعداد را پوشش میدهد و هم دقت مناسبی ارائه میدهد. نکته ی جالب توجه این است که این کاهش دقت حتی در بهبود عملکرد الگوریتم های یادگیری ماشین نیز مؤثر واقع شده است. در شکل زیر مقایسه ی این دو استاندارد را مشاهده میکنید.



در این فاز از پروژه باید یک دستور جدید به نام bfloatConvert را در پردازنده پیاده‌سازی کنید که دو عملوند هشت بیتی میگیرد، عملوند اول قسمت صحیح عدد میباشد و با علامت در نظر گرفته میشود و عملوند دوم قسمت اعشاری عدد میباشد و بی علامت در نظر گرفته میشود. پردازنده باید بتواند sign، exponent و mantissa را محاسبه کند، .سپس exponent را در رجیسترRc ذخیره کند و sign و mantissa را به ترتیب در رجیستر Rb ذخیره کند و عدد تولید شده را نیز در خروجی قرار دهد.

جزئیات پیاده سازی :

1. دستور: bfloatConvert whole, fraction
2. کد دستور: 1101
3. فرمت دستور: قسمت اعشاری_قسمت صحیح_1101

مثال:

نمایش عدد 5.2
دستور ورودی به پردازنده: 1101_00000101_00000010
خروجی: 0100000010100110