گزارش تمرین سه سیستم‌های دیجیتال

متین غیاثی ۴۰۲۱۰۶۲۲۹

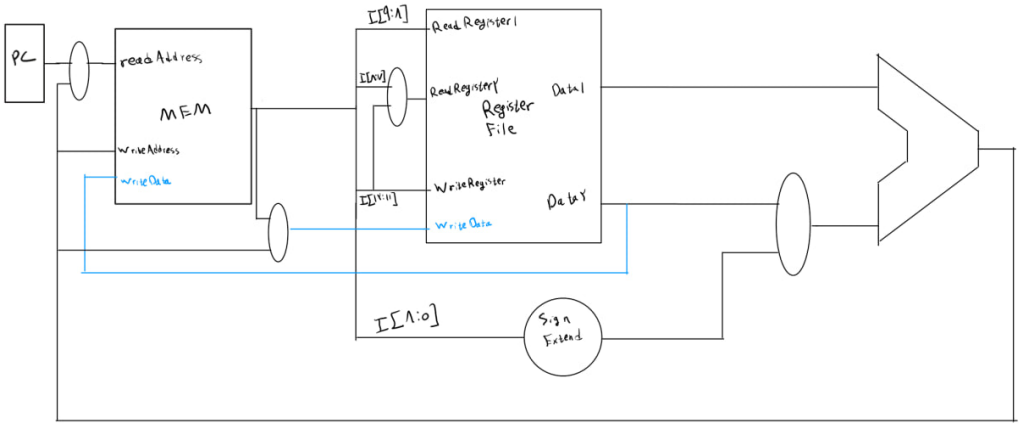
تیر ۱۴۰۴

۱ پیش‌گفتار

در این تمرین یک پردازنده‌ی چندچرخه‌ای (Multi-Cycle) ساده پیاده‌سازی شد. این پردازنده شامل دستورهای ADD، SUB، MUL، DIV، LOAD و STR است. همچنین این پردازنده دارای حافظه به عرض شانزده بیت است که بین حافظه دستورات و داده‌ها مشترک است.

۲ پردازنده

معماری کلی پردازنده به صورت زیر است که ورودی سلکت تسهیم‌کننده‌ها (Multiplexer) توسط واحد کنترل، و بسته به Opcode و وضعیت فعلی تعیین می‌شود.



شکل ۱: شماتیک کلی پردازنده

در پنج تا از دستورات، بیت‌های دوازده و یازدهم فقط برای آدرس نوشتن استفاده می‌شوند اما در دستور STR مقدار آن ثبات باید خوانده، و در حافظه ذخیره شود؛ برای همین، نیاز به وجود یک تسهیم‌کننده قبل از ورودی دوم رجیستر فایل است.
این پردازنده شامل جهار حالت FETCH، DEEX، MEMACC و WB است. در این معماری چون نیازی به اجرای جداگانه‌ی حالت‌های DECODE و EXECUTE با هم ترکیب شدند.
در ادامه به توضیح بخش‌های این پردازنده می‌پردازیم

۱.۲ واحد کنترل

این واحد مسئولیت تعیین حالت فعلی، و همچنین تعیین سیگنال‌های کنترلی را بر عهده دارد. سیگنال‌های کنترلی عبارت‌اند از: **RegWrite**. این سیگنال وظیفه تعیین قابلیت نوشتن بر رجیستر فایل را دارد.

MemWrite. این سیگنال وظیفه تعیین قابلیت نوشتن بر حافظه را دارد.
MemRead. این سیگنال وظیفه قابلیت خواندن از روی حافظه را برعهده دارد.
Alusrc. این سیگنال تعیین می‌کند که ورودی دوم ALU مقدار خروجی دوم رجیستر فایل باشد یا مقدار آنی (Immediate) داده شده در دستور.
MemToReg. این سیگنال تعیین می‌کند که آیا مقدار خوانده‌شده از حافظه به رجیستر فایل نوشته شود یا خیر.
RegRead. این سیگنال تعیین می‌کند که کدام بیت‌های دستور ورودی به عنوان ورودی خواندن دوم رجیستر فایل وارد شود.
وابستگی این سیگنال‌ها به نوع دستور در جدول زیر آمده‌است. اما زمان‌بندی این سیگنال‌ها به حالتی که در آن هستیم بستگی دارد.

RegRead	MemToReg	Alusrc	MemRead	MemWrite	RegWrite	
۰	۰	۰	۰	۰	۱	ADD
۰	۰	۰	۰	۰	۱	SUB
۰	۰	۰	۰	۰	۱	MUL
۰	۰	۰	۰	۰	۱	DIV
۰	۱	۱	۱	۰	۱	LOAD
۱	۰	۱	۰	۱	۰	STR

جدول ۱: سیگنال‌های کنترلی

۲.۲ حافظه

این بخش شامل یک حافظه به عرض ۱۶ بیت است که قابلیت خواندن از روی آن و نوشتن به آن را دارد. این حافظه برای خواندن دستورات و خواندن یا نوشتن داده‌ها مشترک است؛ بنابراین برای تعیین اینکه مقدار مورد نیاز برای خوانده‌شدن، ثبات PC باشد و یا خروجی ALU، باید از یک تسهیم‌کننده استفاده کنیم.

۳.۲ رجیستر فایل

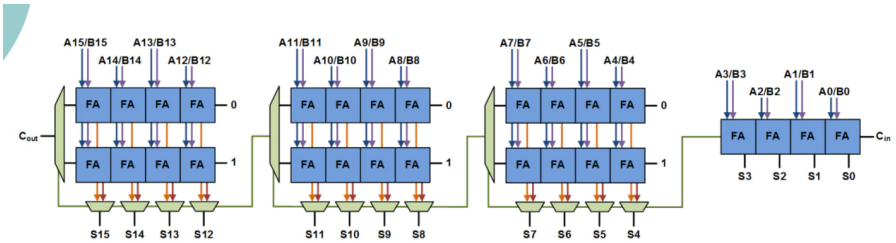
این بخش شامل چهار ثبات x_0 تا x_3 است که همگی این ثبات‌ها General Purpose هستند. رجیستر فایل قابلیت خواندن از دو ثبات و نوشتن به یک ثبات را فراهم می‌کند. همچنین یک سیگنال Write Enable برای این فایل وجود دارد که در صورت صفر بودن، اجازه تغییر مقادیر ثبات را نمی‌دهد.

۴.۲ ALU

این بخش وظیفه انجام محاسبات را دارد. در صورتی که دستور ورودی یکی از دستورات ADD، SUB، MUL یا DIV باشد، خروجی ALU برای نوشتن روی یکی از ثبات‌ها استفاده می‌شود. اما در صورتی که یکی از دستورات LOAD یا STR باشد، خروجی این بخش برای تعیین آدرس استفاده می‌شود. طبق فرمت دستورات، مقدار نوشته‌شده در یکی از ثبات‌ها، با مقدار آنی، بعد از Sign-Extend شدن، جمع می‌شود. تعیین نوع عملیات توسط Opcode تماما مشخص می‌شود. در صورتی که یکی از دستورات R-Type وارد شود، همان کار مد نظر انجام می‌شود. در صورتی که دستور ورودی M-Type باشد، ALU حتما عمل جمع انجام می‌دهد.

۱.۴.۲ جمع

عمل جمع این بخش، به صورت Carry Select Adder است که از اتصال چند Ripple Carry Adder و تسهیم‌کننده ساخته می‌شود. این جمع‌کننده ابتدا در گروه‌های چهار بیتی، برای هر دو حالت مقدار C_{in} حاصل جمع را محاسبه می‌کند.



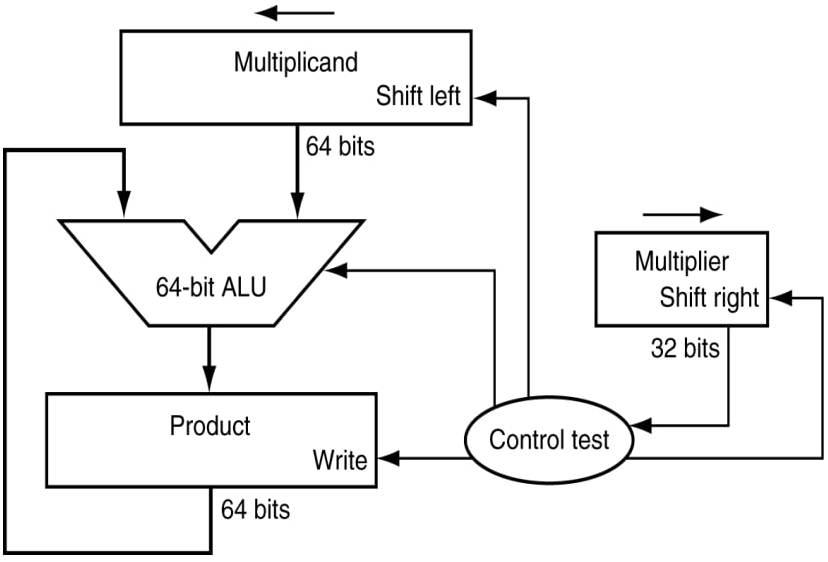
شکل ۲: Carry Select Adder

۲.۴.۲ تفریق

عمل تفریق به صورت جمع ورودی اول با مکمل دوی ورودی دوم انجام می‌شود.

۳.۴.۲ ضرب

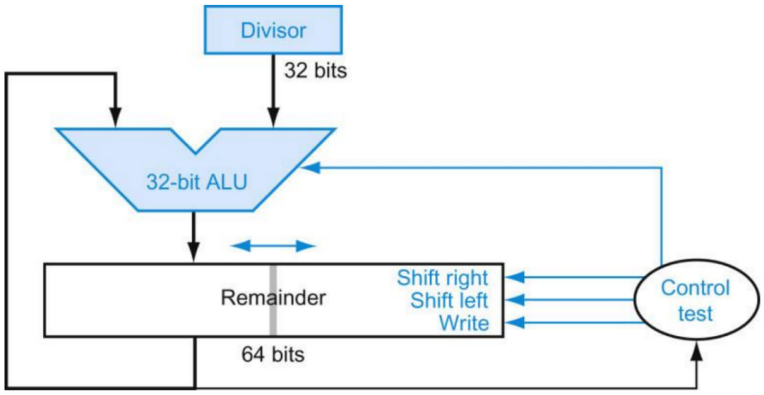
عمل ضرب با استفاده از الگوریتم کاراتسوبا (Karatsuba's Algorithm) انجام می‌شود. این الگوریتم مشابه تقسیم و حل عمل می‌کند. ابتدا دو عدد، به چهار عدد با طول نصف تبدیل می‌شوند، و با سه بار (به جای چهار بار) استفاده از ضرب‌کننده‌ی Shift and Add، و تعدادی عمل جمع، عدد نهایی محاسبه می‌شود. این الگوریتم از پیچیدگی زمانی کمتری نسبت به الگوریتم عادی برخوردار است.



شکل ۳: Shift and Add Multiplier

۴.۴.۲ تقسیم

عمل تقسیم با استفاده از الگوریتم Restoring Division انجام می‌شود. شماتیک کلی این الگوریتم در تصویر زیر آورده شده.



شکل ۴: Restoring Divison Circuit