به‌نام آن‌که جان را فکرت آموخت

دانشگاه صنعتی شریف

دانشکده‌ی مهندسی کامپیوتر

دستور کار

آزمایشگاه معماری کامپیوتر

تهیه و تدوین:

دکتر حمید سربازی‌آزاد

دکتر حسین اسدی

مهندس عطیه غیبی فطرت

تابستان ۱۴۰۳

فهرست عناوین

عنوان صفحه

۱ معرفی آزمایشگاه معماری کامپیوتر

هدف

هدف از آزمایش‌های این جزوه تجربه عملی طراحی و پیاده‌سازی برخی از مفاهیم و روش‌های مطالعه شده در درس معماری کامپیوتر می‌باشد. این جزوه شامل چهار بخش است. بخش اول به آشنایی با ابزار CAD در طراحی و آزمایش درستی عملکرد مدارات منطقی اختصاص دارد. در بخش دوم به طراحی و پیاده‌سازی معماری‌های محاسباتی پرداخته می‌شود. در بخش سوم به طراحی و پیاده‌سازی معماری یک کامپیوتر ساده می‌پردازیم. بخش چهارم به طراحی و پیاده‌سازی همان پردازنده (ساخته شده در بخش سوم) اما با واحد کنترل ریزبرنامه‌پذیر اختصاص دارد.

دانشجویان عزیز در گروه‌های دو یا سه نفری به انجام آزمایش‌های هر بخش می‌پردازند.

پیش‌نیازهای نظری و عملی

درس معماری کامپیوتر و آزمایشگاه مدار منطقی پیش‌نیاز این آزمایشگاه می‌باشند.

تجهیزات و نرم‌افزارهای لازم

در طول این آزمایشگاه، از شبیه‌ساز Proteus برای طراحی مدار و صحت عملکرد آن استفاده می‌شود. همچنین پیاده‌سازی مدار طراحی شده روی بورد در محیط آزمایشگاه انجام می‌شود.

منابع علمی مورد نیاز و چگونگی انجام آزمایش‌ها

تعداد جلسات لازم برای هر آزمایش یک الی دو جلسه (بسته به آزمایش) می‌باشد. دانشجویان مدار مورد نظر در هر آزمایش را طراحی کرده و پس از انتخاب تراشه‌های لازم، آن را پیاده‌سازی و درستی عملکرد آن را بررسی می‌نمایند. برای هر آزمایش، طراحی مدار و انتخاب تراشه‌ها و قطعات لازم جهت پیاده‌سازی قبلاً توسط دانشجویان انجام می‌شود و روز آزمایش به بستن مدار، اشکال‌زدایی و نهایتاً گرفتن نتیجه از مدار اختصاص دارد. هر یک از مدارهای آزمایش‌ها در صورت لزوم باید امکان ورود اطلاعات توسط یک سری کلید (Dip-switch) را داشته و همچنین خروجی را روی نمایشگرهایی از قبیل LED یا 7Segment نمایش دهند.
توصیه می‌شود قبل از بستن مدار نهایی، طرح مورد نظر را در یک محیط شبیه‌ساز آزمایش کرده و از درستی عملکرد آن اطمینان حاصل کنید. سپس مدار نهایی را با استفاده از تراشه‌های مناسب روی بورد در آزمایشگاه پیاده‌سازی کنید. بدین ترتیب، امکان ایجاد تغییرات در مدار و بررسی درستی عملکرد آن به صورت مرحله به مرحله سریع‌تر شده و همچنین تلفات و خرابی تراشه‌ها در

آزمایشگاه کاهش می‌یابد. علاوه بر این، این روش گرفتن نصف امتیاز انجام آزمایش را تضمین می‌نماید.

تهیه گزارش کار آزمایش‌ها

هر گروه موظف است برای هر آزمایش انجام شده گزارشی کامل تهیه کرده و در اولین جلسه آزمایش بعدی به مربی آزمایشگاه تحویل دهد. هر گزارش باید حداقل شامل این موارد باشد: توضیح مختصری درباره مقدمات آزمایش مورد نظر، و سپس بحث و استدلال لازم در انتخاب روش طراحی و پیاده‌سازی ذکر شود. در ادامه، بلوک دیاگرام طرح پیشنهادی (اولیه) آورده شود و پس از آن بین دیاگرام طرح پیشنهادی (اولیه) و دیاگرام طرح نهایی (در صورت تغییر) توضیح علت تغییرات ایجاد شده قرار گیرد.

گزارش‌ها باید طبق اصول ارائه مطالب علمی و فنی در تدوین گزارش‌های دانشجویی تدوین شده باشند. استفاده از کتاب زیر می‌تواند در این خصوص مفید باشد:

عنوان: شیوه ارئه مطالب علمی و فنی
مؤلف: سید محمد تقی روحانی رانکوهی
منتشر کننده: انتشارات جلوه
نوبت چاپ: ششم
سال نشر: ۱۳۸۰

مقررات آزمایشگاه و نحوه ارزیابی

٪۹۰ از نمره هر دانشجو مربوط به انجام دقیق و مرتب آزمایش‌ها و تهیه گزارش توسط گروه است و ٪۱۰ باقی‌مانده به حضور مرتب و به موقع در جلسات آزمایشگاه اختصاص دارد. هر غیبت غیرموجه موجب کسر ۲ نمره از نمره کل آزمایشگاه می‌شود. از ۹۰٪ نمره هر آزمایش، ۶۰٪ مربوط به انجام کامل آزمایش و دریافت نتیجه است و ۳۰٪ مربوط به تهیه گزارش کامل از آزمایش. اگر گروهی موفق به انجام کامل پیاده‌سازی مدار و تست موفقیت‌آمیز آن نشود، انجام آزمایش در محیط شبیه‌ساز ۴۰٪ از ۶۰٪ امتیاز انجام آزمایش را کسب خواهد کرد.

نکته: تمامی اعضای هر گروه باید به آزمایش‌ها مسلط باشند، چرا که نمره شبیه‌ساز Proteus هر فرد به صورت جداگانه در نظر گرفته خواهد شد.

جدول ۱: جدول زمان‌بندی جلسات آزمایشگاه

جلسه	آزمایش	بخش
اول	۱) طراحی و تست مدار جمع دو عدد دهدهی دو رقمی به کمک شبیه‌ساز	**اول:** آشنایی با ابزار CAD
دوم	۲) طراحی و تست یک جمع‌کننده ۶ بیتی با انتخاب رقم نقلی با استفاده از جمع‌کننده‌های ۲ بیتی + تحویل گزارش آزمایش ۱	
سوم	۳) طراحی و پیاده‌سازی ضرب‌کننده ممیز ثابت ۴ بیتی + تحویل گزارش آزمایش ۲	**دوم:** معماری مدارهای محاسباتی
چهارم و پنجم	۴) طراحی و پیاده‌سازی جمع/تفریق‌کننده ممیز شناور + تحویل گزارش آزمایش ۳	
ششم	۵) طراحی و پیاده‌سازی مبدل دهدهی به دودویی + تحویل گزارش آزمایش ۴	
هفتم	۶) طراحی و پیاده‌سازی واحد محاسبه با امکان انتخاب مبدا و مقصد + تحویل گزارش آزمایش ۵	**سوم:** معماری یک کامپیوتر ساده
هشتم	۷) طراحی و پیاده‌سازی واحد محاسبه با امکان کنترل توسط برنامه + تحویل گزارش آزمایش ۶	
نهم و دهم	۸) طراحی و پیاده‌سازی کامل کامپیوتر با حافظه داده و دستورات پرش + تحویل گزارش آزمایش ۷	
یازدهم و دوازدهم	۹) طراحی و پیاده‌سازی مدار کنترل ریزبرنامه‌پذیر + تحویل گزارش آزمایش ۸ در جلسه یازدهم و تحویل گزارش آزمایش ۹ در جلسه دوازدهم.	**چهارم:** معماری ریزبرنامه‌پذیر

۲ آزمایش‌ها

دانشجویان باید در ابتدای هر آزمایش، هدف، شرح و نتایج مورد انتظار آزمایش را مطالعه کرده و قبل از حضور در آزمایشگاه با نحوه انجام آن آشنا باشند. همچنین، طراحی مدار و انتخاب تراشه‌ها و قطعات لازم برای پیاده‌سازی هر آزمایش باید از قبل توسط دانشجویان انجام شود. اگر مطالب تئوری مربوط به آزمایش را فراموش کرده‌اید، لازم است قبل از جلسه آزمایشگاه از منابع مربوطه مطالعه کنید و با آمادگی کامل در جلسه آزمایشگاه حضور یابید.

۲-۱ آزمایش اول: جمع‌کننده دهدهی

هدف

هدف از این آزمایش آشنایی با نحوه عملکرد یک جمع‌کننده دهدهی است. در این آزمایش، دو عدد سه رقمی در مبنای ده به مدار داده می‌شود و نتیجه مورد انتظار در خروجی مشاهده می‌شود (شکل ۱).

شرح آزمایش

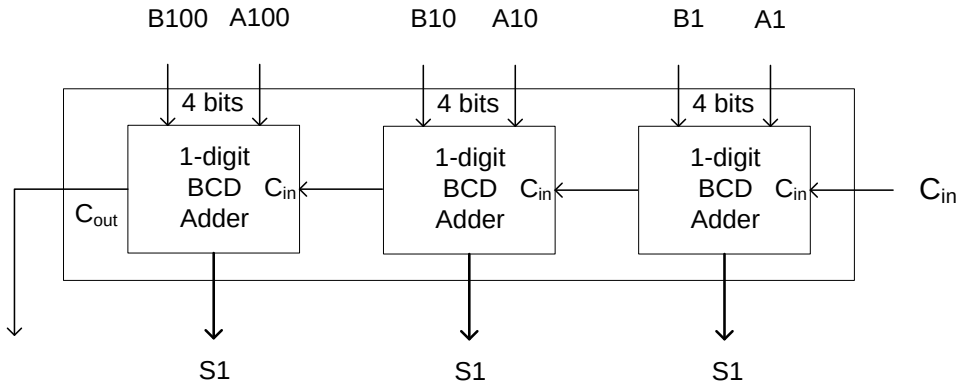
در تمامی آزمایش‌های این آزمایشگاه، استفاده از شبیه‌سازی قبل از پیاده‌سازی نهایی مدار به شدت توصیه می‌شود. این جلسه از آزمایشگاه به آشنایی با یک شبیه‌ساز اختصاص دارد. برای این منظور، لازم است یک مدار جمع‌کننده دهدهی سه رقمی طراحی کرده و عملکرد آن را با کمک شبیه‌ساز بررسی کنید. اغلب برای ساده‌تر شدن کار و راحتی اشکال‌زدایی، طراحی به صورت سلسله مراتبی انجام می‌شود. برای این کار می‌توانید:

- ابتدا یک بلوک تمام جمع‌کننده تک بیتی طراحی کنید.
- سپس با استفاده از بلوک طراحی شده، یک جمع‌کننده دهدهی یک رقمی بسازید.
- نهایتاً، با استفاده از سه جمع‌کننده دهدهی یک رقمی، یک جمع‌کننده سه رقمی طراحی کنید.

پس از طراحی مدار، درستی عملکرد آن را با ورودی‌های مختلف آزمایش کنید.

نتایج مورد انتظار

در این آزمایش، چند سری عدد سه رقمی در مبنای ده به ورودی مدار داده شده و انتظار می‌رود جمع این اعداد در خروجی نمایش داده شود.

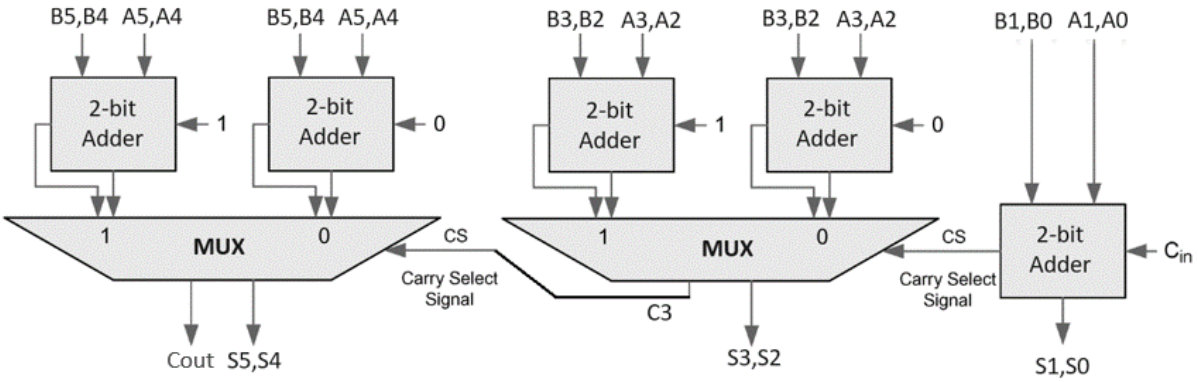


شکل ۱: بلوک دیاگرام جمع‌کننده دهدهی سه رقمی

۲-۲ آزمایش دوم: طراحی Carry Select Adder

هدف

در این آزمایش سعی داریم به منظور بهبود سرعت عمل جمع، یک جمع‌کننده ۶ بیتی با انتخاب رقم نقلی (Carry Select Adder) بسازیم. طبق شکل ۲ ورودی‌ها شامل A[0,5]، B[0,5] و Cin و همینطور خروجی‌ها شامل Cout و S[0,5] می‌باشند.



شکل ۲. ضرب‌کننده ۶ بیتی با انتخاب رقم نقلی (Carry Select Adder)

شرح آزمایش

ابتدا توجه داشته باشید که این جمع‌کننده Hybrid بوده و از جمع‌کننده‌های معمولی ۲ بیتی و Multiplexer تشکیل شده است. این مدار کاملاً ترکیبی است و در هر مرحله، خروجی Carry طبقه قبل به عنوان Selector برای Multiplexer طبقه بعدی استفاده می‌شود.

نتایج مورد انتظار

هر طبقه را به طور مستقل پیاده‌سازی و شبیه‌سازی کنید و سپس طبقات را به هم متصل کرده و نتیجه نهایی را تست کنید. انتظار می‌رود که جمع دو عدد ۶ بیتی با یک بیت Carry ورودی، خروجی ۶ بیتی و یک بیت Carry خروجی را نتیجه دهد.

۲-۳ آزمایش سوم: ضرب‌کننده ممیز ثابت

هدف

در این جلسه یک مدار ضرب‌کننده دودویی چهاربیتی را طراحی و پیاده‌سازی می‌کنیم.

مشخصات مدار مورد نظر به قرار زیر است:

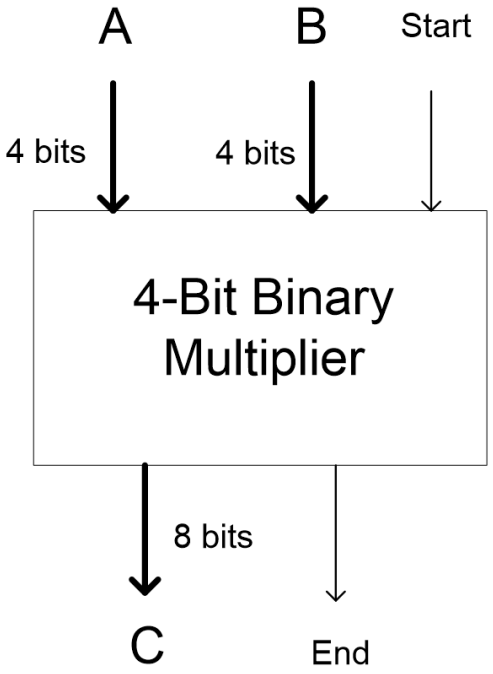
مضروب (ورودی) : A

مضروب‌فیه (ورودی) : B

حاصل‌ضرب (خروجی) : C

شروع ضرب (ورودی) : Start

پایان ضرب (خروجی) : End



شکل ۳. بلوک دیاگرام ضرب‌کننده اعداد صحیح

شرح آزمایش

با فعال شدن سیگنال Start، ضرب‌کننده شروع به کار کرده و حاصل‌ضرب دو عدد ورودی چهار بیتی A و B را به روش shift & add محاسبه می‌کند. پس از اتمام عملیات، حاصل‌ضرب ۸ بیتی را روی خط C قرار داده و با فعال کردن سیگنال End پایان عملیات را اعلام می‌کند.

نکته: هنگام طراحی مدار با استفاده از شبیه‌ساز، سعی کنید از تراشه‌های TTL موجود در کتابخانه شبیه‌ساز و آزمایشگاه استفاده کنید. بدین ترتیب، هنگام پیاده‌سازی عملی، نیازی به تغییر مدار برای استفاده از تراشه‌های موجود نخواهد بود.

نتایج مورد انتظار

در این آزمایش با فعال شدن سیگنال Start، ضرب دو عدد دودویی محاسبه می‌شود. انتظار می‌رود نتیجه صحیح پس از چند سیکل ساعت، همزمان با فعال شدن سیگنال End در خروجی مشاهده شود.

۲-۴ آزمایش چهارم: جمع/تفریق‌کننده ممیز شناور

هدف

در این آزمایش (طی دو جلسه)، مدار یک جمع/تفریق‌کننده ممیز شناور را طراحی کرده و با استفاده ابزار Proteus شبیه‌سازی می‌کنیم. پس از اطمینان از صحت عملکرد در شبیه‌ساز، آن را روی بورد پیاده‌سازی می‌کنیم. مدار اولیه برای شبیه‌سازی را مطابق استاندارد IEEE-754 ۳۲ بیتی طراحی کنید. برای سهولت در پیاده‌سازی روی بورد، تعداد بیت‌ها را از ۳۲ به ۱۲ کاهش دهید.

مشخصات مدار مورد نظر به قرار زیر است:

عملوند اول (ورودی): A

عملوند دوم (ورودی): B

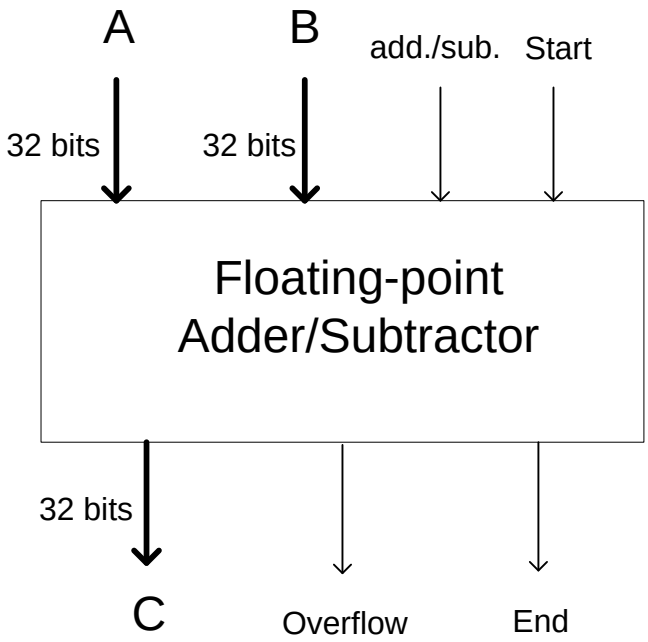
حاصل جمع/تفریق (خروجی): C

شروع عملیات (ورودی): Start

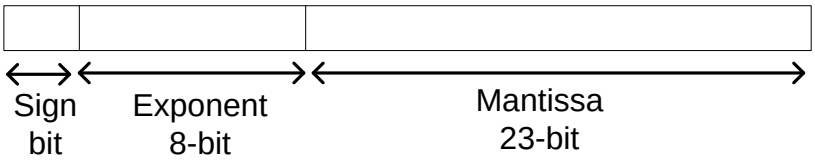
پایان عملیات (خروجی): End

سرریز (خروجی): Overflow

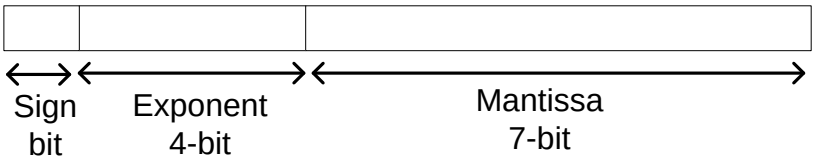
مشخص‌کننده جمع/تفریق (ورودی): add/sub



شکل ۴: جمع/تفریق‌کننده ممیزشناور



شکل ۵: فرمت اعداد ممیز شناور استاندارد IEEE-754 برای طراحی در شبیه‌ساز



شکل ۶: فرمت اعداد ممیز شناور برای پیاده‌سازی بر روی بورد

شرح آزمایش

ابتدا مدار جمع و تفریق کننده دو عدد ممیز شناور (شکل ۴) را با فرمت استاندارد IEEE-754 مطابق شکل ۵ طراحی کرده و با ابزار Proteus شبیه‌سازی نمایید. پس از اطمینان از صحت

عملکرد، طراحی انجام شده را روی بورد پیاده‌سازی نمایید. برای سهولت در پیاده‌سازی، تعداد بیت‌های مدار طراحی شده را مطابق شکل ۶ از ۳۲ بیت به ۱۲ بیت کاهش دهید. با فعال شدن سیگنال Start، مدار شروع به کار کرده و اگر سیگنال add/sub برابر صفر باشد، مقدار A+B و اگر این سیگنال برابر یک باشد، مقدار A-B را محاسبه کرده و روی خط C قرار می‌دهد و سیگنال End را به منزله اتمام عملیات فعال می‌کند. ورودی‌های A و B نرمالیزه بوده و خروجی C نیز باید نرمالیزه باشد. در صورت بروز سرریز، سیگنال Overflow فعال می‌شود.

استفاده از شمارنده با قابلیت شمارش رو به بالا و پایین برای نگهداری نما در طراحی می‌تواند حجم مدار را کاهش دهد.

نتایج مورد انتظار

در این آزمایش، جمع یا تفریق دو عدد دودویی ممیز شناور با فعال شدن سیگنال Start محاسبه می‌شود. انتظار می‌رود نتیجه صحیح بعد از چند سیکل ساعت، بسته به تفاوت دو نما، همزمان با فعال شدن سیگنال End در خروجی مشاهده شود.

۲-۵ آزمایش پنجم: مبدل دهدهی به دودویی

هدف

در این جلسه، مدار یک مبدل دهدهی به دودویی را طراحی کرده و با ابزار Proteus شبیه‌سازی می‌کنیم. پس از اطمینان از صحت عملکرد در شبیه‌ساز، مدار را روی بورد پیاده‌سازی خواهیم کرد.

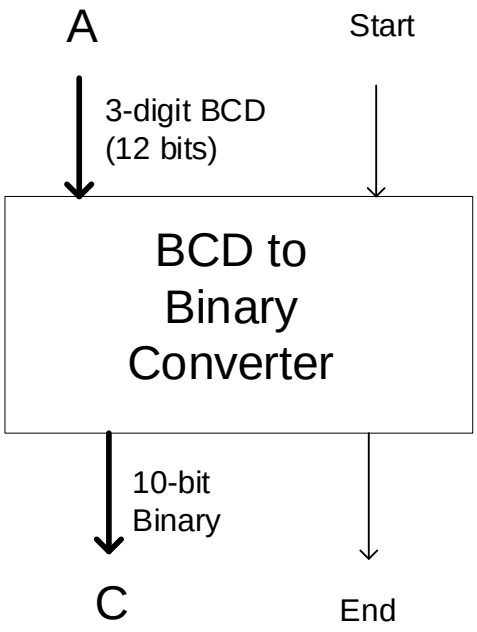
مشخصات مدار مورد نظر به قرار زیر است:

عدد دهدهی (ورودی): A

معادل دودویی (خروجی): B

شروع عملیات (ورودی): Start

پایان تبدیل (خروجی): End



شکل ۷: مبدل دهدهی به دودویی

شرح آزمایش

با فعال شدن سیگنال Start، مدار شروع به کار کرده و ورودی دهدهی که یک عدد سه رقمی است (برای سادگی، در پیاده‌سازی روی بورد اعداد دو رقمی در نظر گرفته شود) را به معادل دودویی آن تبدیل کرده و نتیجه را روی خطوط خروجی قرار می‌دهد و سیگنال End را به عنوان اعلام پایان عملیات فعال می‌کند (شکل ۷).

الگوریتم تبدیل یک عدد دهدهی r رقمی به دودوئی معادل به صورت زیر است:

الف) عدد دهدهی ورودی را یک بیت به راست شیفت دهید.

ب) اگر با ارزش‌ترین بیت رقم iام یک باشد، از آن رقم ۳ تا کم کنید ($1 \le i < r$).

ج) مراحل الف و ب را تا زمانی که تمام ارقام دهدهی صفر شوند تکرار کنید (حداکثر ۱۰ بار تکرار لازم است).

در پایان، بیت‌هایی که با شیفت به راست بیرون می‌آیند، عدد دودویی معادل عدد دهدهی ورودی را تشکیل می‌دهند.

در مثال زیر، عدد دهدهی 110 طبق این الگوریتم به معادل دودویی تبدیل شده است:

عمل	خروجی	رقم ۱	رقم ۲	رقم ۳
شیفت به راست	0	0000	0001	0001
ازرقم‌های 1 و 2 سه تا کم کن	0	1000	1000	0000
شیفت به راست	0	0101	0101	0000
از رقم 1 سه تا کم کن	10	1010	0010	0000
شیفت به راست	10	0111	0010	0000
شیفت به راست	110	0011	0001	0000
از رقم 1 سه تا کم کن	1110	1001	0000	0000
شیفت به راست	1110	0110	0000	0000
شیفت به راست	01110	0011	0000	0000
شیفت به راست	101110	0001	0000	0000
پایان عملیات	1101110	0000	0000	0000

نتایج مورد انتظار

در این آزمایش، با فعال شدن سیگنال Start، انتظار می‌رود عدد دودویی معادل عدد سه رقمی دهدهی ورودی محاسبه شود و پایان کار با فعال شدن سیگنال End مشخص شود.

۲-۶ آزمایش ششم: واحد محاسبه با امکان انتخاب ثبات مبدا و مقصد

هدف

طی آزمایش‌های ششم، هفتم و هشتم یک کامپیوتر ساده را به طور کامل طراحی و پیاده‌سازی کرده و برنامه‌ای را به زبان ماشین نوشته و روی آن اجرا می‌کنیم.

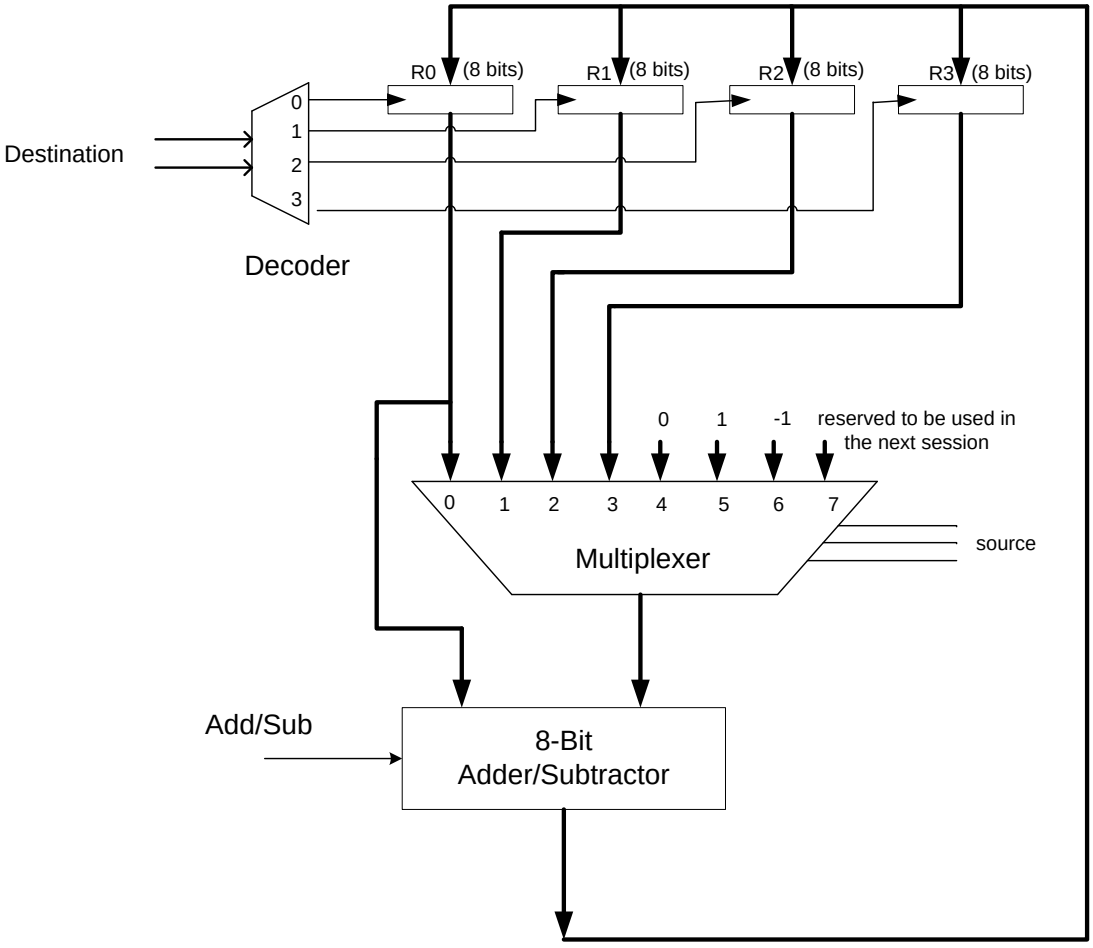
شرح آزمایش

در این آزمایش، واحد محاسبات و مجموعه ثبات‌های عمومی ماشین را طراحی و پیاده‌سازی می‌کنیم. معماری مورد نظر در شکل ۸ نشان داده شده است. این معماری امکان انجام جمع و تفریق با انتخاب ثبات‌های مبدا و ثبات نگهدارنده نتیجه (مقصد) را فراهم می‌کند. چهار ثبات عمومی R0، R1، R2 و R3 هشت بیتی هستند. همانطور که در شکل ۸ نمایش داده شده، یکی از عملوندهای ALU می‌تواند به صورت ثابت محتوای ثبات R0 و دیگری محتوای یکی از ثبات‌های R0 تا R3 یا مقادیر ثابت 0، 1 و 1- باشد. حاصل تولید شده توسط ALU (جمع/تفریق) به یکی از ثبات‌های مقصد R0 تا R3 منتقل می‌شود.

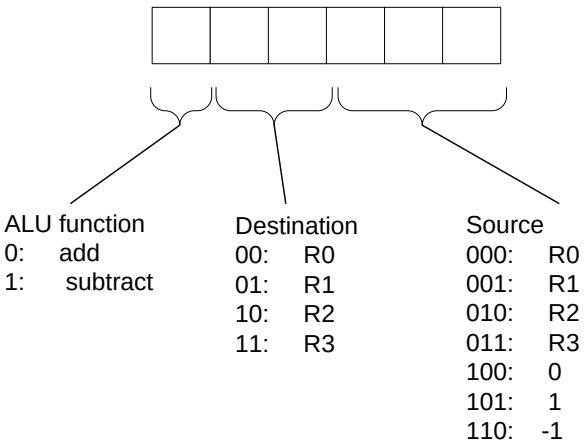
این معماری را به گونه‌ای پیاده‌سازی کنید که قابلیت انجام فرمان‌های شش بیتی در شکل ۹ را داشته باشد.

نتایج مورد انتظار

در این آزمایش، انتظار می‌رود قابلیت اجرای فرمان‌های شش بیتی مشخص شده در شکل ۹ روی ثبات‌های ۸ بیتی ماشین (طبق شکل ۸) ایجاد شود. این قابلیت با اجرای فرمان‌های مختلف مشخص می‌شود.



شکل ۸: معماری واحد محاسبات



شکل ۹: قالب فرمان‌های شش بیتی

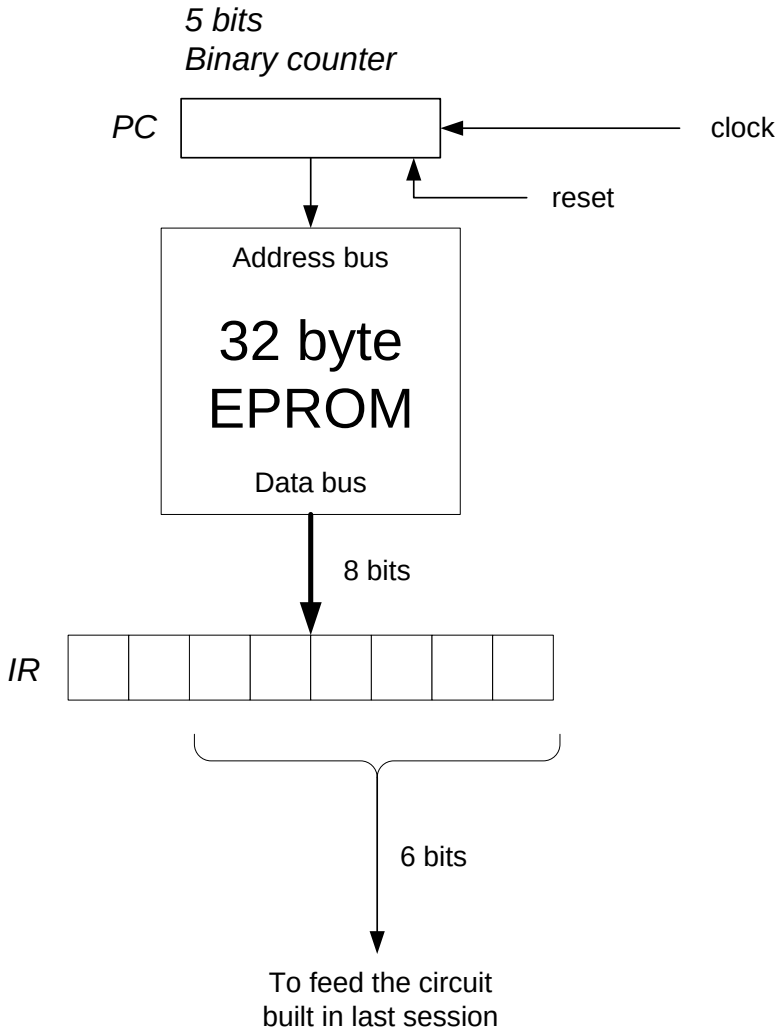
۲-۷ آزمایش هفتم: کنترل توسط برنامه ذخیره شده در حافظه

هدف

هدف از این آزمایش آشنایی با نحوه واکشی دستورات در پردازنده‌ها می‌باشد.

شرح آزمایش

در این آزمایش، فرمان‌های مورد نیاز برای کنترل مدار آزمایش پنجم از برنامه‌ای که در حافظه EPROM ذخیره شده است، گرفته می‌شود. این فرمان‌ها به ترتیب توسط یک شمارنده برنامه (PC) آدرس‌دهی شده و پس از واکشی از حافظه، اجرا می‌شوند. برای این منظور، باید مدارهای لازم به مدار آزمایش ششم افزوده شوند. شکل ۱۰ بلوک دیاگرام سیستم را نشان می‌دهد.



شکل ۱۰: بلوک دیاگرام سیستم

پس از افزودن بخش‌های مورد نیاز به مدار آزمایش پنجم، برنامه زیر را کدنویسی کرده و در حافظه EPROM ذخیره کنید، سپس با استفاده از معماری پیاده‌سازی شده آن را اجرا کنید.

برنامه تولید شش جمله از سری فیبوناچی

در سری فیبوناچی، دو جمله اول 0 و 1 هستند و مقدار هر جمله دیگر برابر با مجموع مقادیر دو جمله قبلی است. سری اعداد فیبوناچی مطابق تابع زیر تولید می‌شود:

$$F(n) := \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F(n-1) + F(n-2) & \text{if } n > 1. \end{cases}$$

قطعه برنامه زیر ده جمله اول این سری را در ثبات‌های R0 و R1 تولید می‌کند.

Address	Code	Instruction	Comment	
00000		Sub R0،R0	R0← 0	جمله اول درR0
		Add R1، 1	R1← 1	جمله دوم در R1
		Add R0،R1	R0←1	جمله سوم در R0
		Add R1،R0	R1←2	جمله چهارم در R1
		Add R0،R1	R0←3	جمله پنجم در R0
		Add R1،R0	R1←5	جمله ششم در R1
		Add R0،R1	R0←8	جمله هفتم در R0
		Add R1،R0	R1←13	جمله هشتم درR1
		Add R0،R1	R0←21	جمله نهم درR0
		Add R1،R0	R1←34	جمله دهم درR1

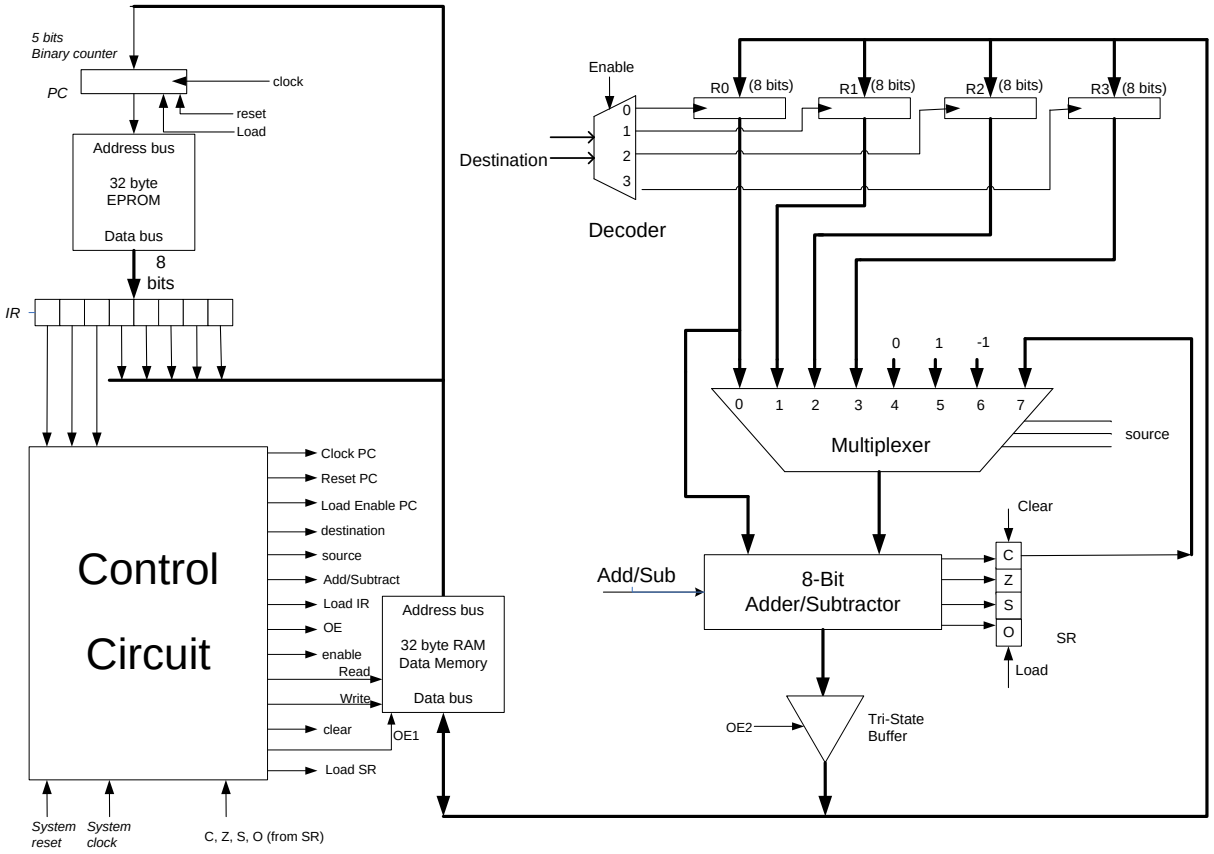
نتایج مورد انتظار

در این آزمایش انتظار می‌رود سیگنال‌های کنترلی از آزمایش قبلی به صورت ترتیبی از یک حافظه واکشی و اجرا شوند.

۲-۸ آزمایش هشتم: استفاده از حافظه داده و دستورات پرش

هدف

در آزمایش هفتم، امکان استفاده از حافظه داده برای ذخیره داده‌های بینابینی را نداشتیم. همچنین، کمبود دستورات پرش و عدم امکان وجود حلقه در برنامه، به وضوح احساس می‌شد. در این آزمایش، مدار آزمایش هفتم را تکمیل کرده و آن را به یک کامپیوتر ساده تبدیل می‌کنیم که دارای امکان دسترسی به حافظه داده برای خواندن و ذخیره داده‌ها و همچنین استفاده از دستورات پرش شرطی و غیرشرطی است.



شکل ۱۱: بلوگ دیاگرام کلی کامپیوتر ساده

شرح آزمایش

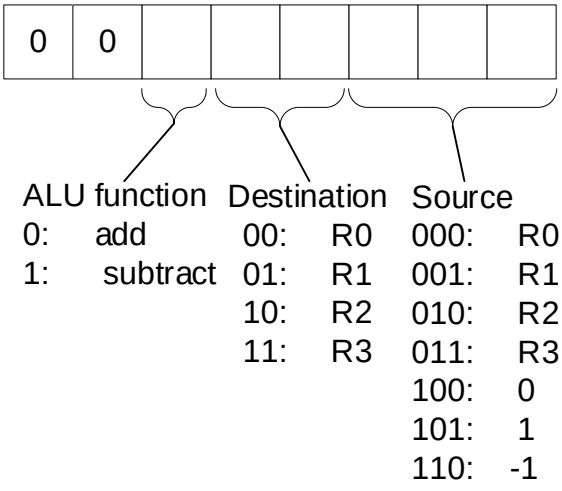
برای حافظه داده‌ها از یک RAM با ظرفیت ۳۲ بایت استفاده می‌کنیم. شکل ۱۱ معماری کامپیوتر مورد نظر را نشان می‌دهد (به این نوع معماری که در آن حافظه برنامه از حافظه داده جداست، معماری Harvard می‌گویند). دستورات این ماشین (به همراه دستورات محاسباتی قابل اجرا در آزمایش هفتم) به سه گروه تقسیم می‌شوند: دستورات محاسباتی – انتقالی، دستورات دسترسی به حافظه داده، و دستورات پرش شرطی و غیر شرطی.

مراحل شبیه‌سازی

آزمایش‌های ۷ و ۸ ابتدا با استفاده از ابزار شبیه‌ساز Proteus پیاده‌سازی شده و نتایج مورد انتظار روی شبیه‌ساز مشاهده می‌شوند. پس از اطمینان از صحت شبیه‌سازی، این مدارها روی بوردهای آزمایشگاهی برنامه‌پذیر پیاده‌سازی شده و نتایج مورد انتظار روی آن‌ها مشاهده می‌شوند. در نهایت، پس از پیاده‌سازی روی این بوردها، آزمایش‌های ۷ و ۸ به صورت فیزیکی روی بوردهای آزمایشگاهی انجام شوند.

دستورات محاسباتی – انتقالی

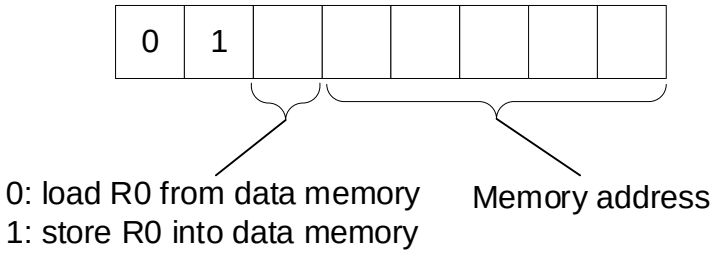
این دستورات شامل عملیات انتقال داده بین ثبات‌ها و عملیات حسابی می‌شوند. قالب این دستورات به صورت نشان داده شده در شکل ۱۲ است.



شکل ۱۲: قالب دستورات محاسباتی-انتقالی

دستورات دسترسی به حافظه داده

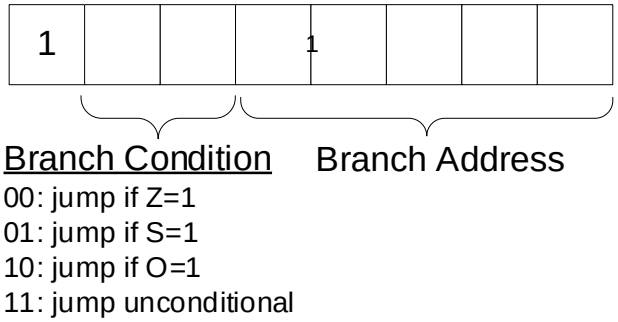
این دسته از دستورات شامل بارگذاری محتوای یک خانه از حافظه به ثبات R0 و ذخیره محتوای ثبات R0 در یک خانه از حافظه است. قالب این دستورات در شکل ۱۳ نشان داده شده است.



شکل ۱۳: قالب دستورات دسترسی به حافظه داده

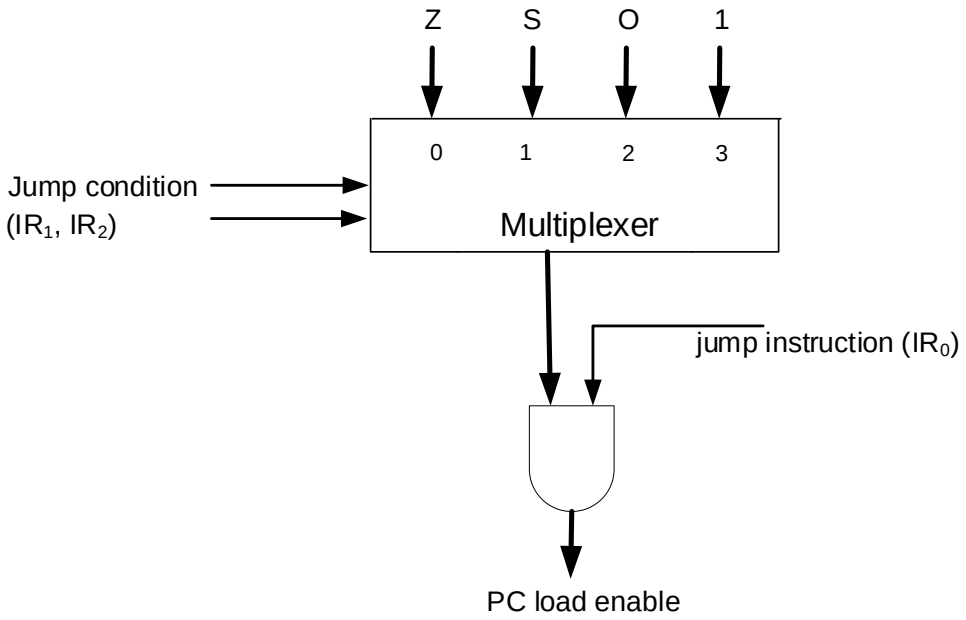
دستورات پرش (شرطی و غیر شرطی)

این دستورات شامل پرش‌های شرطی و غیر شرطی به آدرس دلخواه در حافظه دستور هستند. قالب این دستورات در شکل ۱۴ نشان داده شده است.



شکل ۱۴: قالب دستورات پرش

سیگنال Load شمارنده برنامه (PC) می‌تواند توسط مدار شکل ۱۵ تولید شود.



شکل ۱۵: تولید سیگنال Load شمارنده برنامه

پس از پیاده‌سازی و اطمینان از صحت عملکرد سیستم، برنامه‌ای به زبان ماشین بنویسید که:
الف) مجموع ده جمله اول سری فیبوناچی را محاسبه کرده و در آدرس صفر حافظه داده‌ها ذخیره کند (با استفاده از حلقه).
ب) برنامه‌ای بنویسید که دو عدد ۶۴ بیتی ذخیره شده در آدرس صفر و ۸ حافظه داده را جمع کرده و حاصل ۶۴ بیتی را در آدرس ۱۶ حافظه داده ذخیره کند.

این برنامه‌ها را کدنویسی کرده و در حافظه برنامه ذخیره کنید و سپس اجرا نمایید. لازم است پردازنده پس از اجرای این برنامه‌ها متوقف شود و از پیشروی برای اجرای دستورات بعدی بازماند. برای این منظور، می‌توان در انتهای برنامه پس از آخرین دستور، یک دستور پرش غیرشرطی به آدرس خود دستور پرش اضافه کرد.

نتایج مورد انتظار

در این آزمایش انتظار می‌رود که امکان خواندن از حافظه، نوشتن در حافظه، و اجرای دستورات پرش به آزمایش قبلی افزوده شود. اجرای دو برنامه محاسبه مجموع جملات سری فیبوناچی و جمع دو عدد ۶۴ بیتی نیز مورد انتظار است.

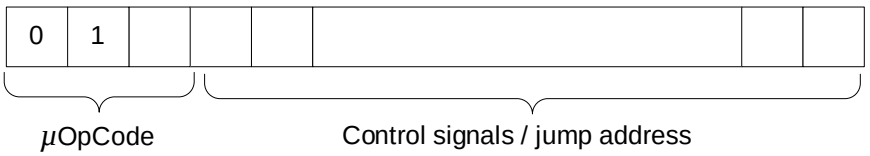
۲-۹ آزمایش نهم: واحد کنترل ریزبرنامه‌سازی شده

هدف

در این آزمایش، مدار کنترل کامپیوتر ساخته شده در آزمایش‌های ششم، هفتم و هشتم را به صورت ریزبرنامه‌پذیر طراحی و پیاده‌سازی می‌کنیم.

شرح آزمایش

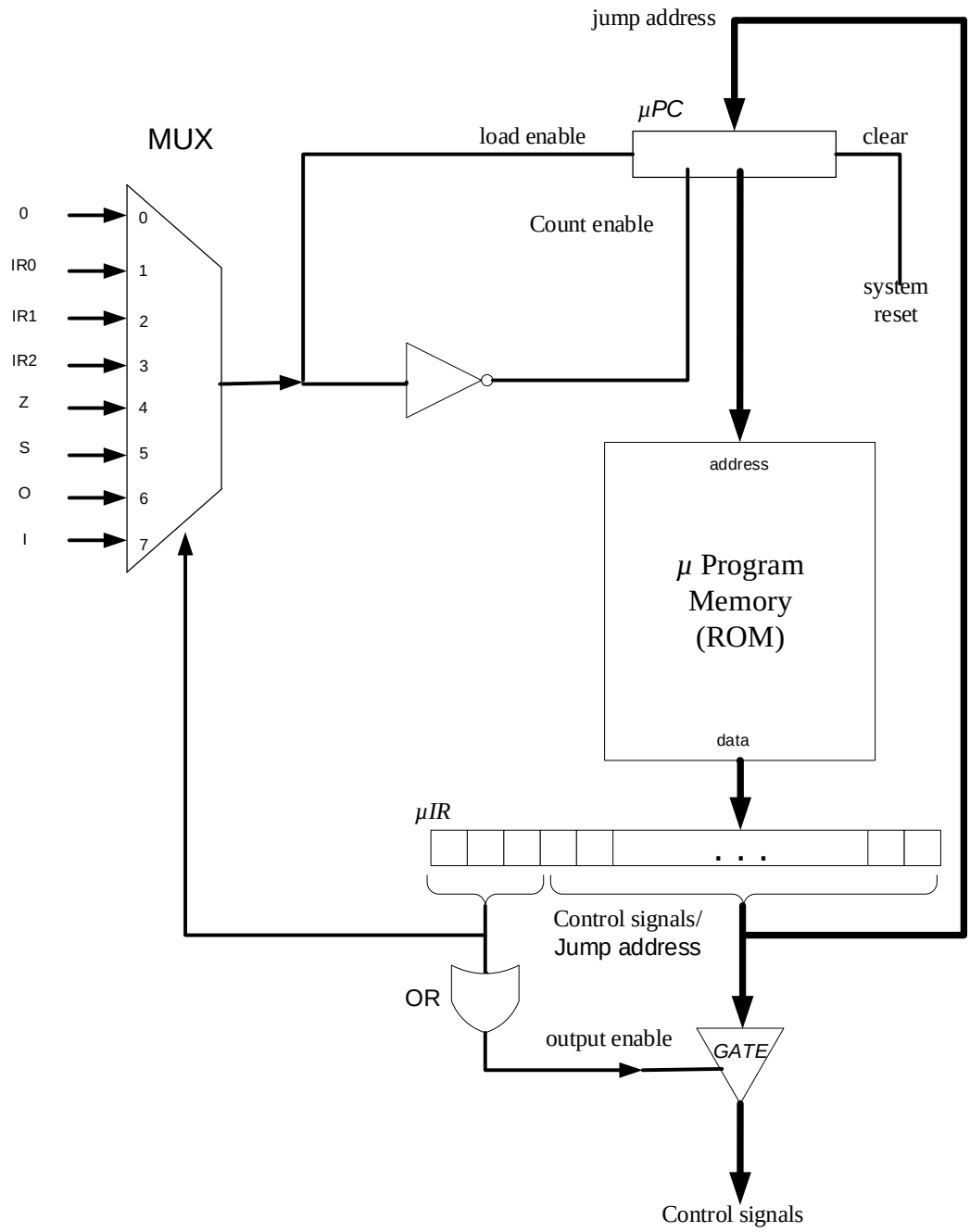
قالب ریزدستورات به صورت نشان داده شده در شکل ۱۶ بوده که شامل ۸ نوع ریزدستور است.



000: normal (no jump)
001: jump if IR0=1
010: jump if IR1=1
011: jump if IR2=1
100: jump if z=1
101: jump if S=1
110: jump if O=1
111: jump always

شکل ۱۶: قالب ریز دستورات

بلوک دیاگرام ریزمعماری اجرای ریزدستورات در شکل ۱۷ آمده است. گنجایش ریزحافظه را ۲۵۶ کلمه فرض کنید. مدار کامل کامپیوتر ساده را که بخش کنترل آن با سیستم ریزبرنامه‌پذیر عمل می‌کند، طراحی و پیاده‌سازی کنید. ابتدا ریزبرنامه‌های مراحل واکشی و اجرای دستورات مختلف ماشین را نوشته و در ریزحافظه ذخیره کنید. پس از اطمینان از صحت عملکرد سیستم، به برنامه الف آزمایش هفتم (جمع ۱۰ جمله اول از سری فیبوناچی) را روی ماشین اجرا کنید.



شکل ۱۷: بلوک دیاگرام ریزمعماری اجرا کننده ریز دستورات

نتایج مورد انتظار

در این آزمایش، انتظار می‌رود که مدار کنترل کامپیوتر ساده به صورت ریزبرنامه‌پذیر طراحی و پیاده‌سازی شود و اجرای دو برنامه جمع جملات سری فیبوناچی و جمع دو عدد ۶۴ بیتی با معماری ریزبرنامه‌پذیر تکرار شود.

منابع

[1] D. Patterson and J. L. Hennessy,
Computer Organization and Design: The Hardware/Software Interface,
6th Edition,
Morgan Kaufmann Publishing,
2020.

[2] M. Mano,
Computer System Architecture,
3rd Edition,
Prentice-Hall,
1992.