دانشکده مهندسی کامپیوتر

آزمایشگاه معماری کامپیوتر

گزارش آزمایش پنجم

عنوان آزمایش : واحد محاسبه با امکان انتخاب ثبات مبدا و مقصد

دکتر حمید سربازی آزاد

سارا آذرنوش — ۹۸۱۷۰۶۶۸

کسری امانی — ۹۸۱۰۱۱۷۱

پارسا محمدیان — ۹۸۱۰۲۲۸۴

۹ شهریور ۱۴۰۰

فهرست مطالب

۱ مقدمه

کامپیوترهای امروزی از بخش‌های مختلفی تشکیل شدند که یکی از بخش‌های اصلی آن‌ها CPU (Central Processing Unit) است که مسئول پردازش است. یکی از بخش‌های اصلی و جدا نشدنی واحد پردازنده مرکزی، ALU (Arithmetic and Logic Unit) است. همانطور که از اسم واحد محاسبه و منطق مشخص است، پردازش‌های محاسباتی در واحد پردازنده مرکزی بر عهده این بخش می‌باشد. ورودی‌های واحد محاسبات با توجه به معماری پردازنده تعیین می‌شوند. برای نمونه پردازنده‌های تک عملونده، دو عملونده، و سه عملونده، داریم که عدد موجود در هر کدام تعداد عملوندهای Explicit یا صریح را مشخص می‌کند. در مقابل عملوند Implicit یا ضمنی در دستور ماشین مشخص نمی‌شود و به صورت پیشفرض در نظر گرفته شده است مانند ماشین انباشتگر یا Accumulator که در آن یک ثبات در نظر گرفته می‌شود و به عنوان مقصد، و یکی از مبداهای محاسبات استفاده می‌شود. حال اگر به شکل ۱ نگاه کنیم، معماری واحد محاسبات به گونه‌ای است که یکی از عملوندهای مبدا به صورت ضمنی R_0 است و دیگر عملوند مبدا و عملوند مقصد به صورت صریح مشخص می‌شوند. در ادامه مشاهده می‌کنیم که در این معماری سه عملوند Immediate نیز وجود دارد. این اعداد، اعداد پرکاربردی هستن که به دلیل کاربرد فراوانشان به صورت غیرقابل تغییر در مدار موجود هستند. همچنین Multiplexer موجود یک ورودی دیگر دارد که در این آزمایش استفاده نمی‌شود ولی بعدها برای Flag استفاده می‌شود. Flagها مجموعه بیت‌هایی هستند که پس از پایان عملیات حسابی بر اساس نتیجه عملیات پر می‌شوند. این پرچم‌ها معمولاً شامل موارد زیر هستند :

۱. Carry : C

۲. Zero : Z

۳. Sign : S

۴. Overflow : O

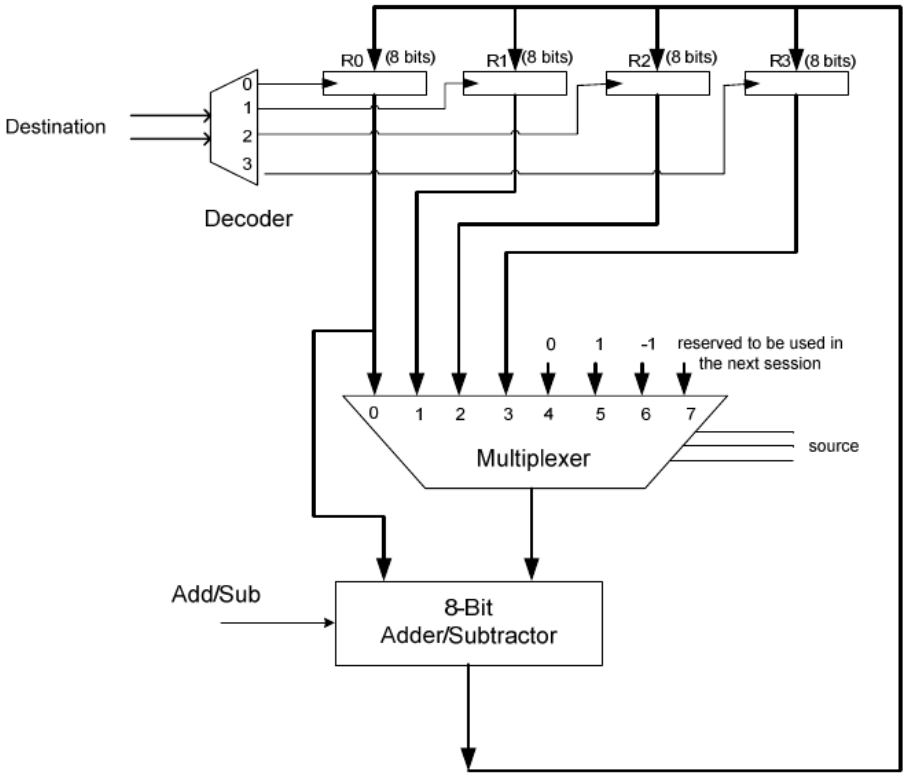
۲ هدف آزمایش

در این آزمایش قصد داریم بخشی از یک کامپیوتر ساده را طراحی کنیم. این بخش شامل یک Register File که درون خود دارای ۴ ثبات ۸ بیتی است، و یک ALU (Arithmetic and Logic Unit) که عملگرهای جمع و تفریق را دارد، است. برای طراحی و پیاده‌سازی این بخش از نرم‌افزار Proteus استفاده می‌کنیم.

۳ قطعات مورد نیاز

قطعات مورد استفاده در این آزمایش در زیر آمده اند :

- گیت XOR
- گیت NOT
- گیت OR
- گیت AND



شکل ۱: معماری واحد محاسبات

- LOGICPROBE
- LOGICPROBE (BIG)
- LOGICSTATE
- SWITCH
- BUTTON
- 74S139 IC → 2 to 4 Decoder
- 74198 IC → 8-Bit Register
- 3WATT10K Resistor
- 74157 → Quadruple 1-of-2 Multiplexer

۴ شرح آزمایش

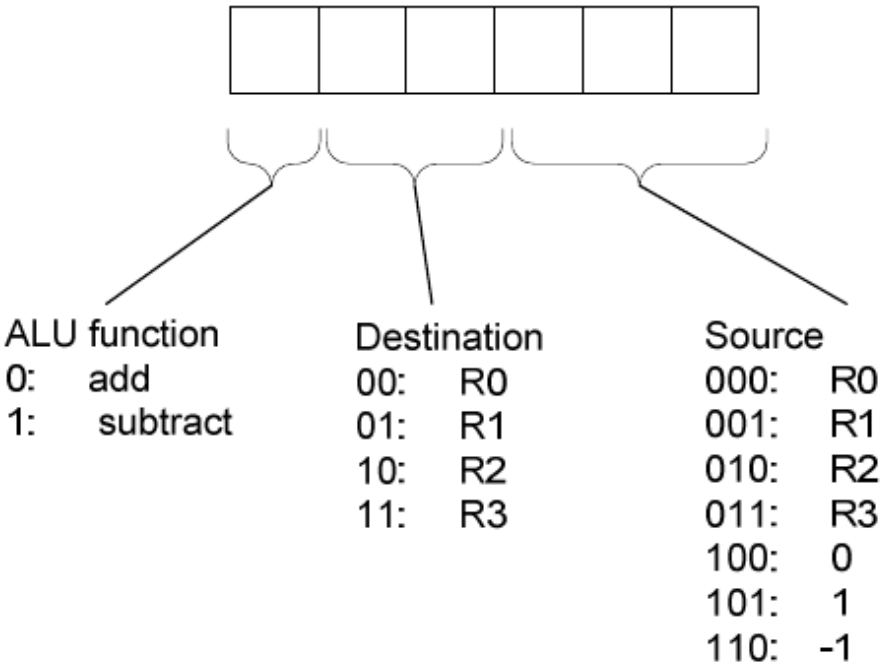
همانطور که گفته شد قسمت محاسباتی یک CPU ساده را طراحی می‌کنیم. برای ساخت مدار از طراحی موجود در شکل ۱ استفاده می‌کنیم. این معماری امکان انجام جمع و تفریق با انتخاب یکی از ثبات‌های مبدا و ثبات نگهدارنده نتیجه (مقصد) را دارد.

۱.۴ قالب فرمان‌های واحد محاسبات

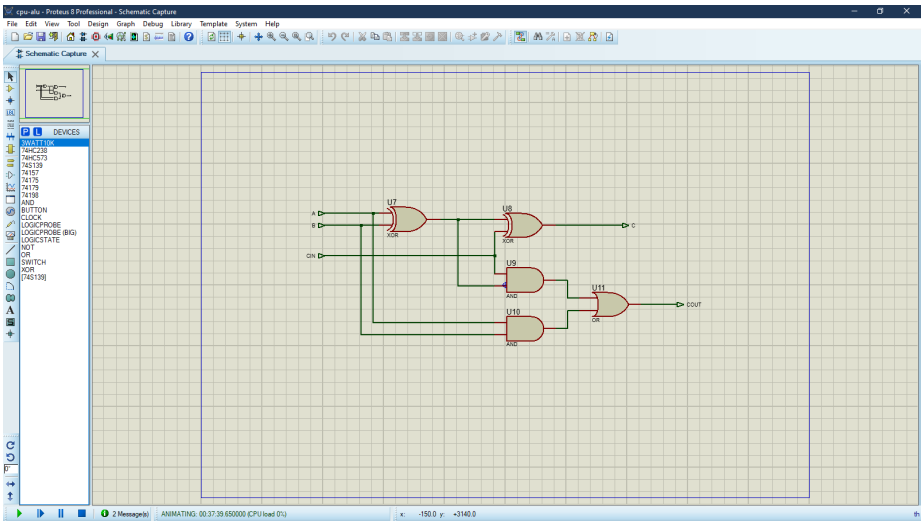
دستورات این واحد محاسبات ۶ بیتی هستند و فرمت دستورات در شکل ۲ آمده است.
عملوندها در ۴ رجیستر ۸ بیتی R_0 تا R_3 قرار دارند. مقدار هر یک از رجیسترها به یک Multiplexer با ۸ ورودی ۸ بیتی وارد می‌شود که با توجه به سه بیت آخر دستور ورودی مناسب به خروجی وصل می‌شود و وارد Adder/Subtractor می‌گردد. همچنین ورودی دیگر آن نیز به صورت ضمنی ثبات R_0 است.
با توجه به بیت اول دستور، نوع عملیات (جمع/تفریق) مشخص می‌گردد (اگر ۰ باشد جمع و اگر ۱ باشد تفریق است).
خروجی Adder/Subtractor به همه ثبات‌ها متصل است. پایه‌ی لود کردن موازی ثبات مطلوب به وسیله یک Decoder و توسط دو بیت وسط دستور فعال می‌گردد و در نهایت نتیجه محاسبات در آن دخیره می‌شود.

۲.۴ طراحی Adder/Subtractor

با توجه به ۸ بیتی بودن اعداد نیاز به یک جمع/تفریق کننده ۸ بیتی داریم. برای ساخت آن مانند آزمایش قبل عمل می‌کنیم. یک جمع کننده کامل (Full Adder) با گیت های پایه می‌سازیم. مدار تمام جمع‌کننده در شکل ۳ آمده است.
سپس با استفاده از ۸ جمع کننده کامل و اتصال متوالی آن‌ها می‌توانیم Ripple-Carry Adder بسازیم.
می‌دانیم عملیات تفریق در مبنای دو همان عملیات جمع با مکمل ۲ یک عدد است. پس برای اضافه کردن عملیات تفریق، یک بیت عملیات به عنوان ورودی ماژول در نظر می‌گیریم، در صورتی که ۱ بود تمام بیت‌های عدد دوم را معکوس می‌کنیم و کل عدد را با یک جمع می‌کنیم. عملیات معکوس کردن ۸

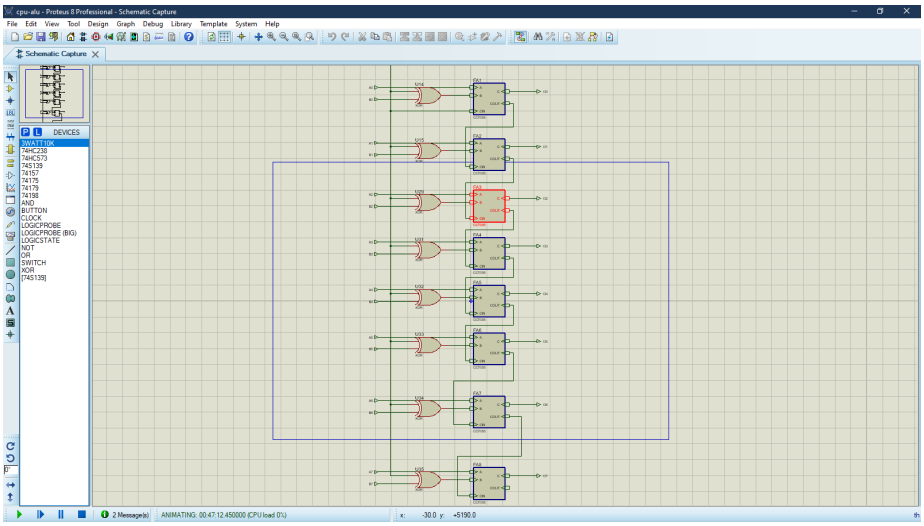


شکل ۲: فرمت دستورات واحد محاسبه



شکل ۳: مدار تمام جمع‌کننده

بیت عدد دوم با استفاده از ۸ گیت XOR انجام شده است و عملیات جمع کردن با یک، با پاس دادن رقم نقلی اولیه انجام شده است. به این صورت هر وقت func یک باشد عملیات تفریق (جمع با مکمل دو) انجام می‌شود. مدار این جمع/تفریق کننده در شکل ۴ قابل مشاهده است.



شکل ۴: مدار جمع/تفریق کننده

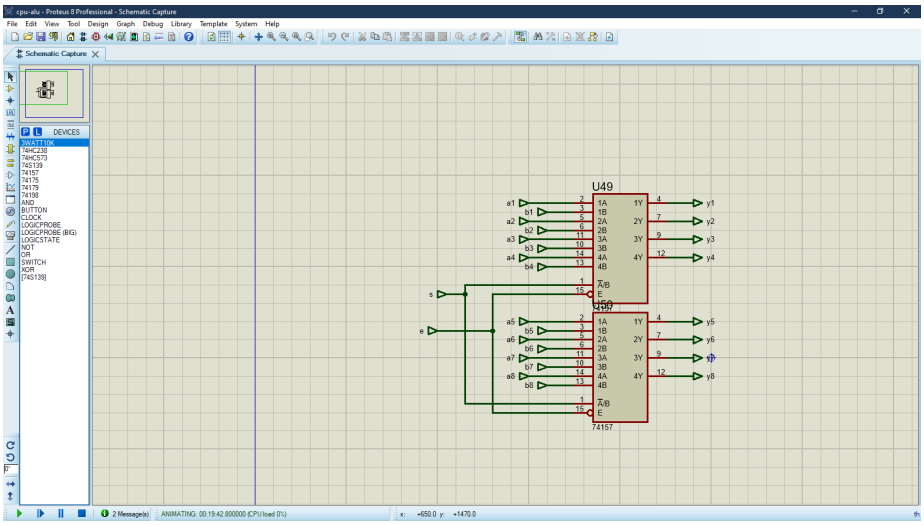
۳.۴ طراحی مولتی‌پلکسر با ۸ ورودی ۸ بیتی

برای ساختن مولتی‌پلکسر با ۸ ورودی ۸ بیتی، ابتدا ماژول مولتی‌پلکسری با ۲ ورودی ۸ بیتی می‌سازیم. این ماژول با استفاده از دو آی سی 74157 ساخته شده است. پیاده‌سازی مدار این ماژول را در شکل ۵ مشاهده می‌کنیم.

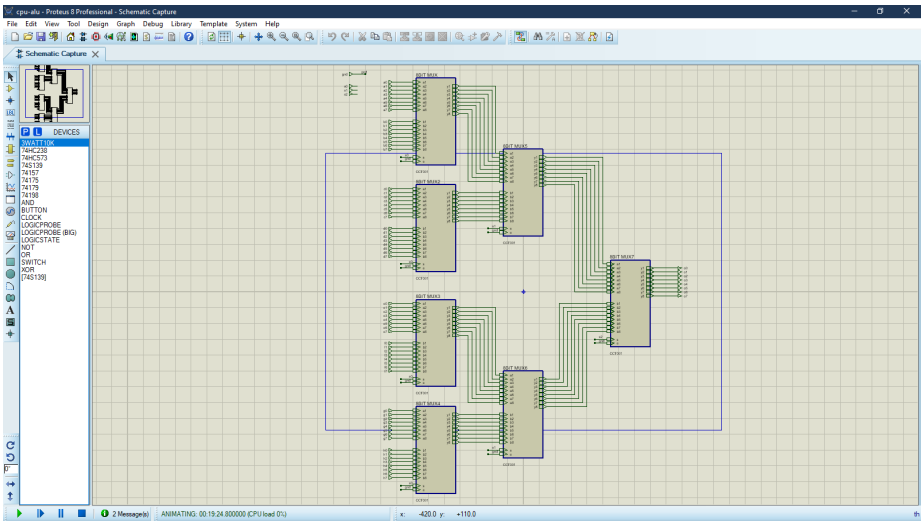
حال هفت عدد از ماژول ساخته شده را متوالی وصل می‌کنیم تا مولتی‌پلکسر مطلوب را بسازیم. این اتصال به گونه‌ای است که از ۳ لایه ماژول ساخته شده تشکیل شده است. لایه اول بیت S_0 به ماژول‌ها داده می‌شود، لایه دوم بیت S_1 و لایه آخر بیت S_2. مدار نهایی این ماژول نیز در شکل ۶ قابل مشاهده است.

۴.۴ کلید Enable

مدار با بستن کلید Enable فعال می‌شود و یا توجه به دستور داده شده، شروع به کار می‌کند. کلید Enable از یک سمت به مقاومت و مقدار ۱، و از سمت دیگر به زمین متصل است. با بستن کلید چون به زمین وصل می‌شود مقدار آن ۰ می‌شود و با استفاده از نات آن مقدار Enable یک می‌شود و مدار فعال می‌شود. دقت شود که این سیگنال در پردازنده نیاز نیست و صرفا برای انجام تست اضافه شده است. چون در پردازنده در هر سیکل ساعت پردازش انجام می‌شود.



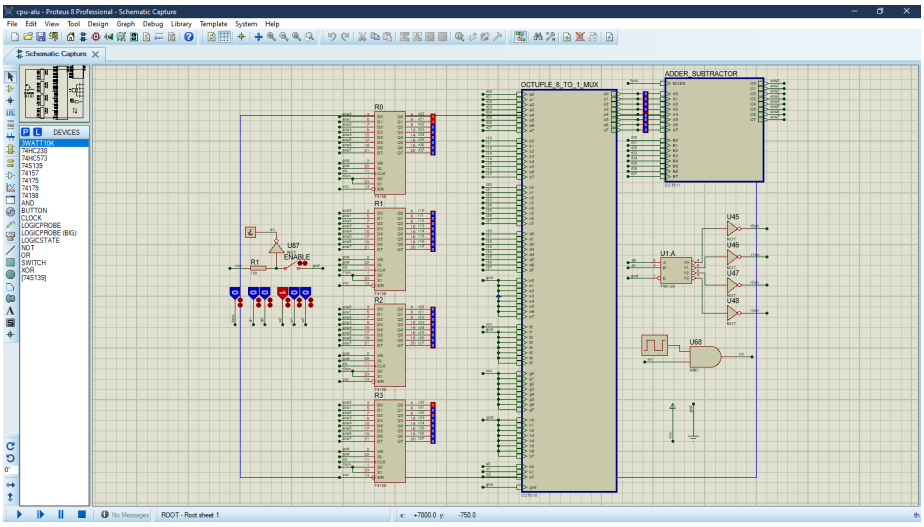
شکل ۵: مدار درونی مولتی‌پلکسر



شکل ۶: مدار مولتی‌پلکسر

۵ نتیجه آزمایش

در آخر مدار یک واحد محاسبات را داریم که قابلیت انتخاب ثبات یکی از ورودی‌ها و خروجی را دارد. مدار نهایی در شکل ۷ قابل مشاهده است.

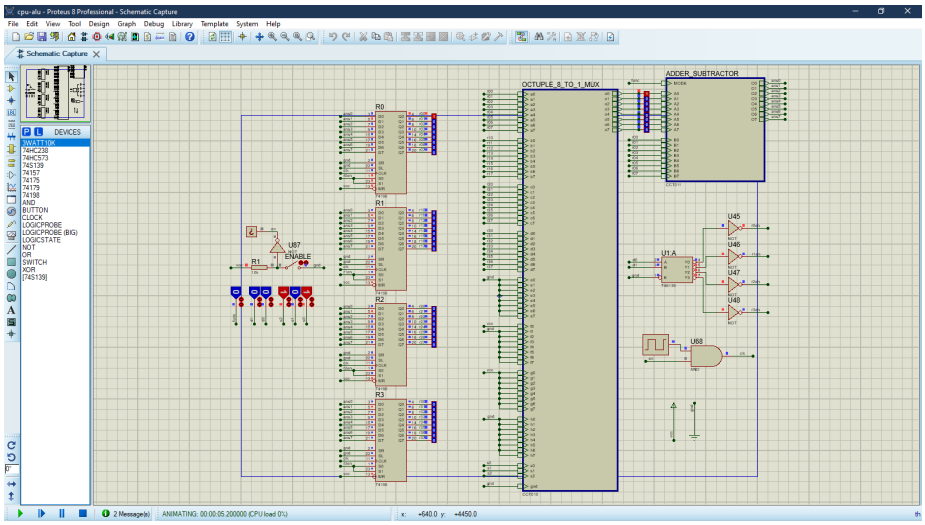


شکل ۷: مدار نهایی

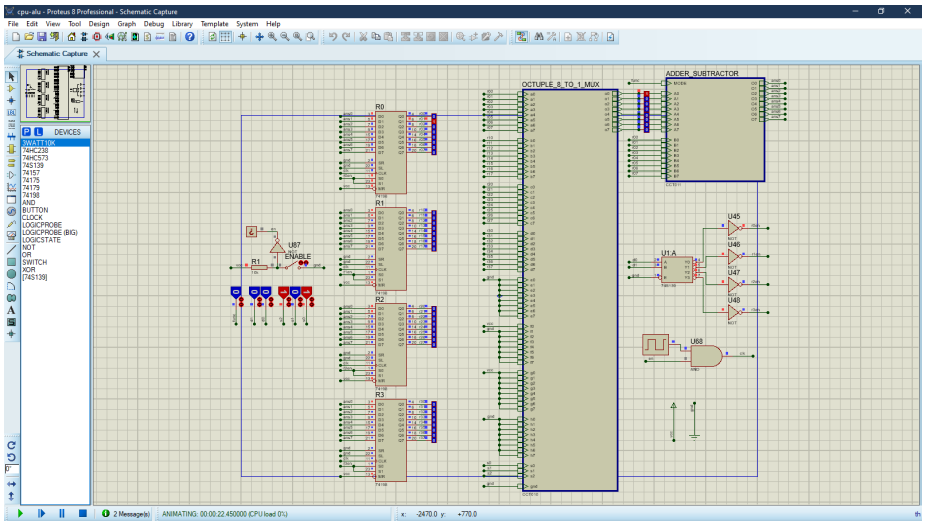
برای تست مدار چندین ورودی مختلف را تست می‌کنیم.

۱.۵ تست

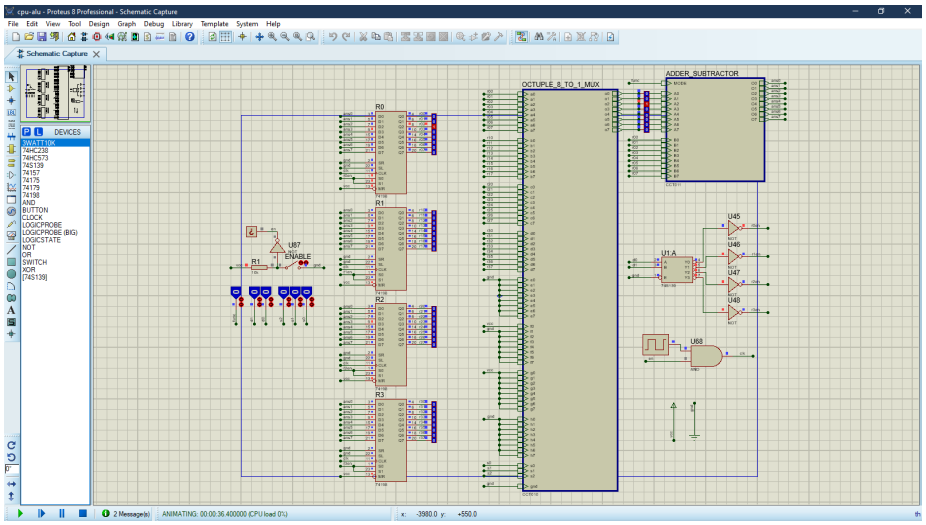
در این تست ابتدا دو بار $R_0 = R_0 + 1$ انجام شده که در شکل ۸ و ۹ قابل مشاهد است. سپس $R_0 = R_0 + R_0$ انجام شده است که در شکل ۱۰ قابل مشاهده است. پس از آن $R_1 = R_0 + R_0$ انجام شده است که در شکل ۱۱ نمایان است. بعد از آن $R_2 = R_0 + R_0$ اجرا شده است که در شکل ۱۲ مشخص است. سپس $R_3 = R_1 + R_0$ اجرا شده است که در شکل ۱۳ آمده است. در آخر برای اطمینان از صحت عملگر تفریق $R_3 = R_3 - R_0$ اجرا شده است که در شکل ۱۴ نتیجه مشاهده می‌شود.


cpu-alu - Proteus 8 Professional - Schematic Capture
OCTUPLE_8_TO_1_MUX
ADDER_SUBTRACTOR
R0
R1
R2
R3
U87
ENABLE
U45
U46
U47
U48
U58
U1:A

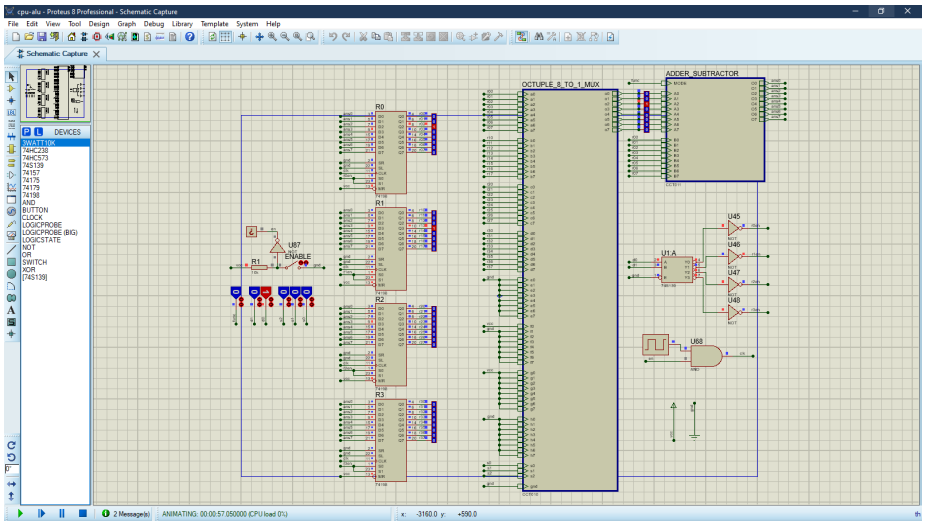
شکل ۸


cpu-alu - Proteus 8 Professional - Schematic Capture
OCTUPLE_8_TO_1_MUX
ADDER_SUBTRACTOR
R0
R1
R2
R3
U87
ENABLE
U45
U46
U47
U48
U58
U1:A

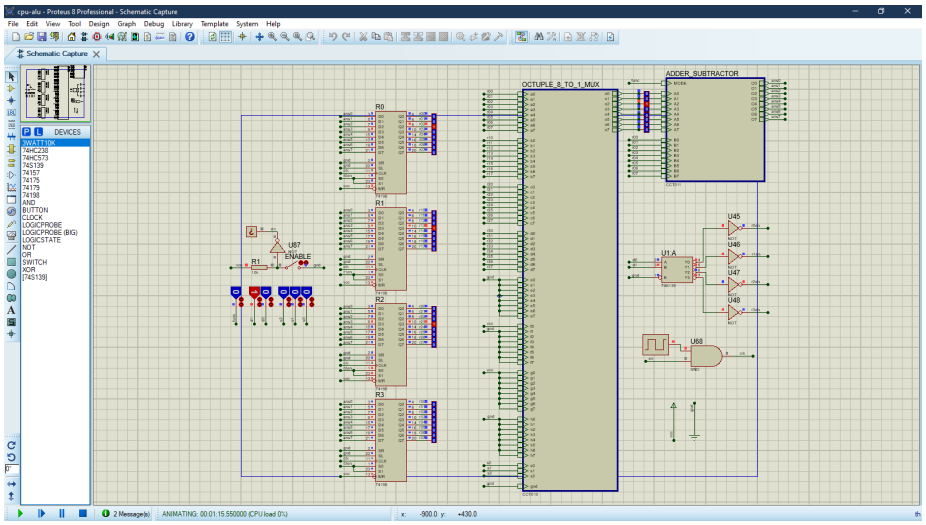
شکل ۹


cpu-alu - Proteus 8 Professional - Schematic Capture
OCTUPLE_8_TO_1_MUX
ADDER_SUBTRACTOR
R0
R1
R2
R3
U87
ENABLE
U45
U46
U47
U48
U58
U1:A

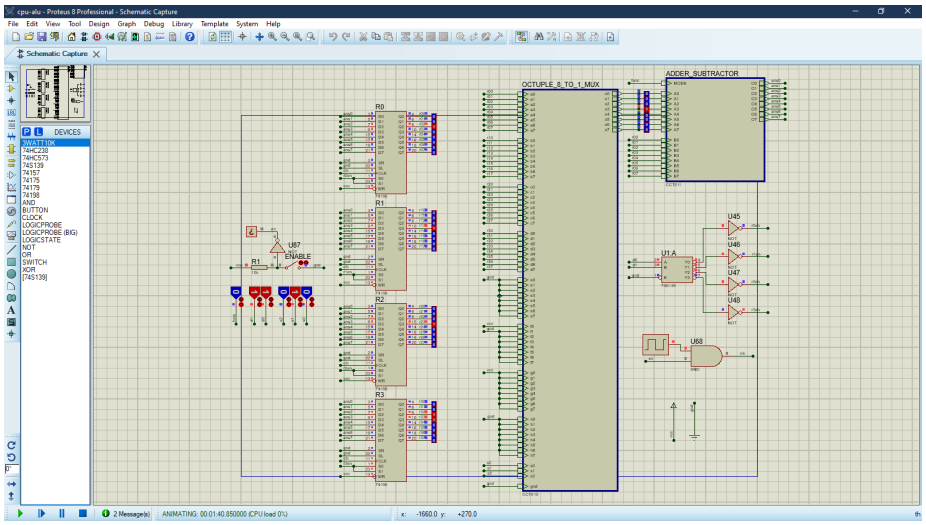
شکل ۱۰


cpu-alu - Proteus 8 Professional - Schematic Capture
OCTUPLE_8_TO_1_MUX
ADDER_SUBTRACTOR
R0
R1
R2
R3
U87
ENABLE
U45
U46
U47
U48
U58
U1:A

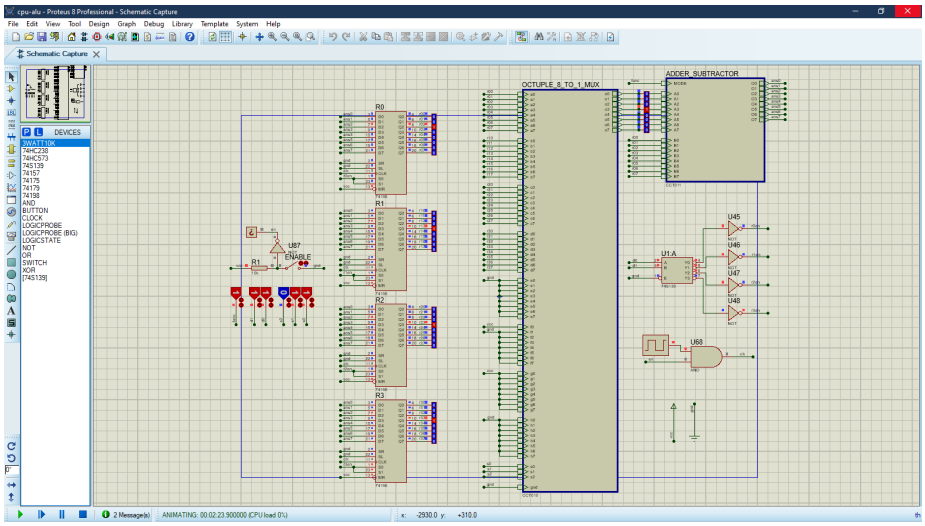
شکل ۱۱


cpu-alu - Proteus 8 Professional - Schematic Capture
OCTUPLE_8_TO_1_MUX
ADDER_SUBTRACTOR
R0
R1
R2
R3
U87
ENABLE
U45
U46
U47
U48
U58
U1:A

شکل ۱۲


cpu-alu - Proteus 8 Professional - Schematic Capture
OCTUPLE_8_TO_1_MUX
ADDER_SUBTRACTOR
R0
R1
R2
R3
U87
ENABLE
U45
U46
U47
U48
U58
U1:A

شکل ۱۳


cpu-alu - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
DEVICES
74HC238
74HC573
74S139
74157
74175
74179
74198
AND
BUTTON
CLOCK
LOGICPROBE
LOGICPROBE (BIG)
LOGICSTATE
NOT
OR
SWITCH
XOR
[74S139]
OCTUPLE_8_TO_1_MUX
ADDER_SUBTRACTOR
R0
R1
R2
R3
U87
ENABLE
U45
U46
U47
U48
U1:A
U68
2 Message(s)
ANIMATING: 00:02:23.900000 (CPU load 0%)

شکل ۱۴