پروژه‌ی نهایی

- مهلت تحویل: چهارشنبه 14/04 ساعت 23:55
- زمان تحویل حضوری: پنج‌شنبه 15/04
- پروژه‌ی خود را در قالب یک فایل zip با نام ID_Project (که ID شماره‌ی گروه شما است) در صفحه‌ی درس در CourseWare (CW) بارگذاری کنید.
- برای کاهش حجم فایل نهایی می‌توانید فولدرهای db و simulation را از فایل زیپ نهایی که در سایت بارگذاری می‌شود حذف کنید.
- برای پروژه باید یک گزارش کامل شامل تمام جزئیات مورد نظر برای ساخت پردازنده به صورت تایپ شده آماده کرده و ضمیمه‌ی تمرین نمایید. گزارش بخشی از نمره‌ی پروژه‌ی شما را شامل می‌شود. در گزارش خود روند طراحی تمامی مراحل طراحی را به طور کامل توضیح دهید.
- پروژه در قالب گروه‌های حداکثر سه نفره قابل انجام است. در گزارش پروژه، بخش‌هایی که توسط هر یک از اعضای گروه انجام شده است باید مشخص شود. در تحویل حضوری، هر فرد باید به بخش خود تسلط کامل داشته و از بخش‌های انجام شده توسط سایر اعضا هم آگاهی داشته باشد. انتظاری که از گروه‌های سه نفره می‌رود بیشتر از گروه‌های دو و یک نفره است، لذا سعی کنید در گروه خود عنصر free-rider نداشته باشید. هر گونه شباهت بی‌دلیل به پروژه‌های گروه‌های دیگر تقلب محسوب خواهد شد. در تحویل حضوری پروژه حضور تمامی اعضای گروه الزامی است.
- مشخصات اعضای گروه را در این لینک وارد کنند.
- سوالات خود را صرفاً در فروم مربوطه در CW بپرسید.

طراحی پردازنده خط لوله

در این پروژه هدف طراحی و آزمایش عملکرد یک پردازنده‌ی خط لوله است.

- استفاده از تمامی ماژول‌های طراحی شده در تمرین‌های قبلی اعضای گروه مجاز است.
- استفاده از هر تعداد ثبات میانی با هر اندازه‌ای در پروژه مجاز است. دقت کنید مانند تمارین قبل تمامی ثبات‌ها باید ساخته‌ی خودتان باشد و همچنان استفاده از ماژول‌های آماده‌ی lpm، یا استفاده از ثبات‌های ساخته شده توسط دیگر دانشجویان مجاز نیست.
- تمامی ثبات‌های پردازنده، به همراه بانک ثبات اصلی باید دارای یک درگاه ریست سنکرون باشند که به سیگنال ریست پردازنده وصل شده است و با یک شدن آن، در لبه‌ی کلاک تمامی ثبات‌ها مقدار صفر به خود می‌گیرند.
- در این تمرین برای پیاده‌سازی حافظه، چهار فایل در پوشه‌ی memory پیوست شده است. این فایل‌ها را به پوشه‌ی پروژه‌ی خود منتقل کرده و از ماژول آن‌ها در پروژه استفاده نمایید.
- مانند تمرین‌های درس، طراحی ماژولار و قابل فهم، بخش مهمی از نمره‌ی شما را در بر می‌گیرد.

مشخصات ماژول حافظه

- دو حافظه‌ی مجزا برای حافظه‌های داده و دستور با نام‌های instmemory و datamemory در نظر گرفته شده است.
- این حافظه‌ها دارای یک درگاه آدرس اصلی با نام addr به طول 16 بیت و یک درگاه اصلی ورودی داده به نام datain و به طول 32 بیت است. همچنین یک درگاه یک بیتی با نام write نیز مشخص کننده‌ی عمل خواندن (write = 1) و یا عمل نوشتن (write = 0) خواهد بود.
- این ماژول دارای 2^{16} ردیف 32 بیتی از اعداد است. بدین ترتیب آدرس‌دهی به حافظه با اعداد 16 بیتی امکان‌پذیر است و پهنای هر خط از حافظه 32 بیت است.
- برای مقداردهی اولیه به این حافظه، فایل با پسوند .v را مشاهده نموده و داده‌ی مربوط به آدرس مورد نظر را تغییر دهید. همچنین می‌توانید برای خانه‌هایی که مقداردهی نشده‌اند خط مربوطه را به کد اضافه نمایید.
- درگاه reset حافظه به صورت سنکرون و حساس به لبه‌ی مثبت است.
- سیگنال‌های پرچم که در تمرین ساخت ALU طراحی شدند، همچنان در این پردازنده به همان حالت حضور داشته و استفاده می‌شوند.
- بانک ثبات استفاده شده در این پردازنده، مشابه تمرین‌های قبلی شما شامل 32 ثبات 32 بیتی است.

بررسی صحت عملکرد

برای بررسی صحت عملکرد این پردازنده، باید دو برنامه‌ی مجزا بنویسید و فایل حافظه‌ی شامل این دو برنامه را به همراه پروژه بارگذاری نمایید.

- برنامه‌ی اول، ضرب ماتریس: این برنامه برای ضرب دو ماتریس پنج در پنج است. ماتریس اول به صورت row-order major از خانه‌ی 0 تا 24 حافظه، و ماتریس دوم از خانه‌ی 25 تا 49 حافظه ذخیره شده است که باید به دلخواه به آن مقدار دهید. در نهایت نتیجه‌ی ضرب دو ماتریس باید به صورت یک ماتریس 5 در 5 از خانه‌ی 50 تا 74 حافظه ذخیره شود.
- برنامه‌ی دوم، مرتب‌سازی آرایه: این برنامه به منظور مرتب‌سازی عناصر یک آرایه به طول 100 است که در خانه‌ی 0 تا 99 حافظه ذخیره شده است. اعضای آرایه باید به صورت صعودی از خانه‌ی 100 تا 199 حافظه ذخیره شوند. الگوریتم مورد استفاده در مرتب‌سازی دلخواه است.

- به منظور تایید صحت عملکرد پردازنده، نیاز خواهد بود تا مقدار ثبات‌های میانی خط لوله را به عنوان خروجی پردازنده هم در نظر بگیرید تا بتوان مقادیر آن‌ها را حین اجرای پردازنده توسط فایل Waveform بررسی کرد.
- پردازنده‌ی طراحی شده باید قابلیت اجرای forwarding هایی شامل ALU-ALU و Mem-ALU را داشته باشد.
- نکته‌ی مهم: لازم است خروجی حافظه‌ی داده به عنوان خروجی اصلی پردازنده در نظر گرفته شود و پس از اتمام برنامه، داده‌های مورد نظر در سیکل‌های متوالی از حافظه خوانده و نمایش داده شوند.

نمرات امتیازی

- زمان اجرای برنامه (تعداد سیکل‌های کلاک) نسبت به گروه‌های دیگر کمینه باشد. برای این منظور لازم است که الگوریتم‌ها و دستورات استفاده شده با دقت انتخاب شوند. همچنین لحظه‌ی اتمام برنامه هم باید مشخص شود. در گزارش خود تعداد سیکل‌های اتمام برنامه را ذکر کنید.
- پیاده‌سازی روش‌هایی مانند branch delay slot نیز منجر به زمان اجرای کمتر برنامه خواهد شد. پیاده‌سازی این روش‌ها (حتی اگر زمان اجرای برنامه نسبت به بقیه کمینه نشود) نمره‌ی تشویقی خواهد داشت.
- پردازنده در شبیه‌سازی timing هم درست کار کند.

دستورات مورد استفاده در این پردازنده در ادامه به طور کامل توضیح داده شده است.

دستورات نوع 1:

Opcode					Reg1					Reg2					Reg3					Reg4					Shift Amount					Unused	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Opcode	Operation	Description
00001	Add	R1 = R2 + R3
00010	Subtract	R1 = R2 − R3
00011	AND (Bitwise)	R1 = R2 & R3
00100	OR (Bitwise)	R1 = R2 | R3
00101	Shift Left Logical	R1 = R2 << Shift Amount

00110	Shift Right Logical	R1 = R2 >> Shift Amount
00111	Max	R1 = Max (R2 , R3)
01000	Set on Less Than	if (R2 < R3) then R1 = 1 , else R1 = 0
01001	Multiply	{R1 , R2} = R3 × R4
01010	Move	R1 = R2
01011	Add Indirect	R1 = Mem[R2] + Mem[R3]
10111	Multiply High	R1 = {R2 × R3} [63:32]
11000	Multiply Low	R1 = {R2 × R3} [31:0]
11001	Single Add Indirect	R1 = Mem[R2] + R3

- در دستوراتی که از ثبات برای آدرس‌دهی به حافظه استفاده شده است، مانند Mem[R1]، 8 بیت کم‌ارزش این ثبات (R1[7:0]) لحاظ می‌شود.
- در دستور Set on Less Than، در صورت برقرار بودن شرط، ثبات مقصد مقدار 1 را به خود می‌گیرد و تمام بیت‌های آن 1 نمی‌شود.
- برای عمل ضرب از ماژول lpm_mult می‌توانید استفاده کنید.

دستورات نوع 2:

Opcode					Reg1					Reg2					Immediate																		Unused
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Opcode	Operation	Description
01100	Load Upper Immediate	R1[31:16] = Imm , R1[15:0] = 0x0000
01101	Add Immediate	R1 = R2 + SE(Imm)
01110	OR Immediate	R1 = R2 | Imm
01111	Set on Less Than Immediate	if (R2 < Imm) then R1 = 1 , else R1 = 0

- منظور از عبارت SE که در دستور Add Immediate استفاده شده است Sign Extend است. دقت شود اعداد به شیوه‌ی two's complement نمایش داده می‌شوند، بنابراین عملیات Sign Extend با توجه به بیت پرارزش عملوند انجام خواهد گرفت.
- در دستور OR Immediate، 16 بیت پرارزش ثبات مقصد برابر با 16 بیت پرارزش ثبات مبدا می‌شود و 16 بیت کم‌ارزش، حاصل OR 16 بیت کم‌ارزش ثبات مبدا و عدد Immediate خواهد شد.

دستورات نوع 3:

Opcode					Reg1					Reg2					Address									Unused									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Opcode	Operation	Description
10000	Branch if Equal	if (R1 == R2) then PC ← Address
10001	Branch if Not Equal	if (R1 != R2) then PC ← Address
10010	Jump Register	PC ← R1[7:0]
10011	Jump	PC ← Address
10100	Load Word	R1 = Mem[Address]
10101	Add Memory	R1 = R2 + Mem[Address]

10110	Store Word	Mem[Address] = R1
10110	Store Word	Mem[Address] = R2
11010	Load Indirect	R1 = Mem[R2]
11011	Store Indirect	Mem[R2] = R3

ADDNDR $1 $2 $3

LWNDR $1 $2

SADDNDR $1 $3