فاز اول پروژه:

در این مرحله از پروژه می‌بایست در محیط شماتیک نرم‌افزار Quartus، پردازنده‌ای با مشخصات زیر را پیاده سازی کنید.

- فرمت و مجموعه **دستورالعمل‌های محاسباتی و منطقی**:

Opcode						Destination Reg					Source Reg 1					Source Reg 2					S/R Amount					Reserved					
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	*Instruction*	*Description*
000001	ADD	DST ⟵ SRC1 + SRC2
000010	SUB	DST ⟵ SRC1 – SRC2
000011	MUL	DST ⟵ SRC1 × SRC2
000100	DIV	DST ⟵ SRC1 ÷ SRC2
000101	MOD	DST ⟵ SRC1 % SRC2
000110	MAX	DST ⟵ MAX (SRC1, SRC2)
000111	MIN	DST ⟵ MIN (SRC1, SRC2)
001000	NOT	DST ⟵ ~SRC1
001001	NAND	DST ⟵ SRC1 ~& SRC2
001010	XNOR	DST ⟵ SRC1 ~^ SRC2
001011	SHL	DST ⟵ SRC1 << S/R AMOUNT
001100	SHRL	DST ⟵ SRC1 >> S/R AMOUNT
001101	ROL	DST ⟵ ROTATE LEFT (SRC1, S/R AMOUNT)
001110	ROR	DST ⟵ ROTATE RIGHT (SRC1, S/R AMOUNT)
001111	SLT	SRC1<SRC2: DST = 1

- فرمت و مجموعه **دستورالعمل‌های با عملوند صریح (Immediate):**

Opcode						Destination Reg					Source Reg					Immediate Data													
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	*Instruction*	*Description*
010000	LDI	DST[15:0] ⟵ IMM
010001	LUI	DST[31:16] ⟵ IMM
010010	ADDI	DST ⟵ SRC + SIGN EXTEND (IMM)
010011	SUBI	DST ⟵ SRC – SIGN EXTEND (IMM)
010100	MULI	DST ⟵ SRC × SIGN EXTEND (IMM)
010101	DIVI	DST ⟵ SRC ÷ SIGN EXTEND (IMM)
010110	NANDI	DST ⟵ SRC ~& SIGN EXTEND (IMM)
010111	XNORI	DST ⟵ SRC ~^ SIGN EXTEND (IMM)

- فرمت و مجموعه **دستورالعمل‌های دسترسی به حافظه:**

Opcode						Value Reg (VR)					Address Reg (AR)					Offset													
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	*Instruction*	*Description*
011000	LW	VR ⟵ MEM [$AR+ SIGN EXTEND (Offset)]
011001	SW	MEM [$AR+ SIGN EXTEND (Offset)] ⟵ VR
011010	LB	VR[7:0] ⟵ MEM [$AR+ SIGN EXTEND (Offset)]
011011	SB	MEM [$AR+ SIGN EXTEND (Offset)] ⟵ VR[7:0]

- فرمت و مجموعه **دستورالعمل‌های پرش و توقف:**

Opcode						Reg 1					Reg 2					Address													
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	*Instruction*	*Description*
011100	JMP	PC ⟵ PC[31:18] | Address | “00”
011101	JR	PC ⟵ $Reg1
011110	BEQ	REG1 == REG2: PC ⟵ PC + SIGN EXTEND (Address | “00”)
011111	BLT	REG1 < REG2: PC ⟵ PC + SIGN EXTEND (Address | “00”)
000000	HLT	STOP PC

توضیحات:

- طول کلمه در این معماری ۳۲ بیت است.
- تعداد ثبات‌های عمومی ۳۲ است.
- کوچکترین واحد دسترسی به حافظه بایت بوده و فضای آدرس دهی از صفر تا $۲^{۳۲}-۱$ است. اما **می‌توانید برای تست و سنتز مدار خود در Quartus، سایز حافظه را کوچکتر در نظر بگیرید.**
- دستورالعمل‌ها به صورت **خط لوله**[1] اجرا می شوند.
- نمایش اعداد در این معماری به صورت مکمل ۲ است.
- برای طراحی هر یک از اجزاء مورد نیاز می‌توانید در صورت نیاز از بلوک‌های اساسی در قسمت Wizard استفاده کنید.
- تمام ثبات‌ها (اعم از PC، IR و ...) می‌بایست یک درگاه ریست ناهمگام[2] داشته باشند که به سیگنال ریست پردازنده متصل باشند (با یک شدن این سیگنال محتوای این ثبات‌ها صفر می‌شود).
- در دستورالعمل‌های BLT و SLT مقایسه مقدار دو ثبات به صورت بی‌علامت[3] انجام می‌شود.
- در دستورات LB، SB و LDI بخش های باقی مانده باید به صورت SIGN EXTEND پر شوند. اما در دستور LUI بخش کم ارزش که عمل لود در آن صورت نمی گیرد، باید بدون تغییر باقی بماند.
- نتیجه تابع mode که نشانگر باقی مانده تقسیم SRC1 بر SRC2 است، در مواردی که هر دو عدد منفی باشند، ممکن است منفی باشد. همچنین، در صورتی که دو عدد هم علامت نباشند، فرض کنید نتیجه تقسیم همواره منفی است در نتیجه، ممکن است باقی مانده منفی باشد.
- از دستور HLT برای اعلام پایان اجرای برنامه به پردازنده استفاده می شود. در نتیجه، با اجرای این دستور، نباید PC دیگر افزایش پیدا کند تا فراخوانی دستور بعدی از حافظه متوقف شود.
- برای طراحی ساده تر دستورات ضرب و تقسیم می توانید از مدار ترکیبی به جای ترتیبی استفاده کنید. در صورت تمایل به استفاده از مدار ترتیبی برای این دستورات می توانید درخت تقسیم کلاک را به گونه ای طراحی کنید که از ورود سیگنال کلاک به بقیه واحدها تا آماده نشدن نتیجه این دستورات، جلوگیری کنید.

---

[1] Pipeline
[2] Asynchronous
[3] Unsigned

- انواع مخاطراتی[4] که ممکن است در پردازنده‌ی طراحی شده رخ دهد را گزارش کرده و برای رفع آن‌ها در صورت نیاز ساختار پردازنده را تغییر دهید.
- پردازنده‌ی خود را توسط برنامه‌ای که تمام دستورالعمل‌های نام برده در آن وجود داشته باشد، بیازمایید.
- پس از اطمینان از صحت اجرای تمامی دستورالعمل‌ها، کد ماشین محاسبه جمله *n*ام دنباله فیبوناچی را داخل حافظه قرار دهید و نتیجه اجرای برنامه روی پردازنده و زمان اجرا بر اساس تعداد سیکل را گزارش کنید.
- حال فرض کنید یک حافظه نهان[5] به اندازه‌ی ۴ کیلوبایت بین پردازنده و حافظه اصلی سیستم قرار گیرد. همچنین، فرض کنید پردازنده همواره داده‌ی مورد نیاز خود را از حافظه نهان می‌گیرد و در صورت فقدان داده در حافظه نهان، ابتدا بلوک حاوی داده موردنظر از حافظه‌ی اصلی به حافظه نهان آورده شده و سپس، داده از حافظه‌ی نهان به پردازنده تحویل داده می‌شود. لازم به ذکر است انتخاب تمامی پارامترهای طراحی حافظه‌ی نهان از جمله ساختار حافظه (نگاشت مستقیم، نگاشت انجمنی کامل یا نگاشت انجمنی مجموعه‌ای)، اندازه‌ی بلوک، مکانیزم جایگزینی بلوک‌های حافظه نهان و سیاست نوشتن اختیاری می‌باشد.
- پس از اطمینان از صحت اجرای تمامی دستورالعمل‌ها در پردازنده‌ی جدید، کد ماشین محاسبه جمله *n*ام دنباله فیبوناچی را مجددا داخل حافظه قرار دهید و تعداد سیکل کاهش یافته در ساختار جدید را گزارش کنید.
- استفاده از کد Verilog یا VHDL تنها برای طراحی واحدهای کنترلی مجاز می‌باشد.

فاز دوم پروژه:

در این مرحله، می‌بایست از بین پروژه‌های **الف** تا **د** یکی را به دلخواه انتخاب کرده و کمک پردازنده‌ی متناظر با آن را در محیط شماتیک نرم‌افزار Quartus پیاده سازی کنید. سپس، باید پردازنده‌ی کمکی خود را به پردازنده‌ی اصلی که در فاز قبل طراحی کرده‌اید، متصل کنید تا پردازنده‌ی اصلی بتواند از این واحد برای اجرای دستورالعمل‌های کمکی استفاده کند. جزئیات طراحی کمک پردازنده، نحوه‌ی اتصال پردازنده‌ها و طراحی ساختار دستورالعمل‌ها برای کمک پردازنده بر عهده دانشجویان است. توجه داشته باشید که ساختار دستورالعمل‌های کمک پردازنده باید مطابق با فرمت ساختار دستورالعمل‌های پردازنده‌ی اصلی باشد. همچنین، برای مدیریت ارتباط بین پردازنده‌ی اصلی و پردازنده‌ی کمکی می‌توانید به تعداد مورد نیاز، دستور به مجموعه

[4] Hazard
[5] Cache

دستورالعمل‌های پردازنده‌ی اصلی اضافه کنید. درنهایت، باید مشخص گردد که هر یک از عملیات خواسته شده در چند سیکل انجام می شود.

الف) کمک پردازنده برای محاسبات برداری: این کمک پردازنده باید قادر به انجام دستورالعمل‌های جمع، تفریق، ضرب نقطه‌ای، ضرب عدد در بردار، تقسیم بردار بر عدد، نرمال کردن و محاسبه‌ی اندازه روی بردارها به صورت predicate باشد (یعنی عملیات برداری روی زیرمجموعه‌ای از عناصر بردار که توسط ماسک[6] مشخص می‌شود انجام شود). همچنین، می‌توانید در صورت نیاز دستورالعمل‌های دسترسی به حافظه، کنترلی و پرشی برای کمک پردازنده در نظر بگیرید. کمک پردازنده باید بتواند یک بردار را از حافظه خوانده و آن را در حافظه ذخیره نماید. پس از اتمام مراحل طراحی، کمک پردازنده‌ی خود را توسط برنامه‌ای که تمام دستورالعمل‌های نام برده در آن وجود داشته باشد، بیازمایید. پس از اطمینان از صحت اجرای تمامی دستورالعمل‌ها، با استفاده از دستورات اسمبلی برنامه‌ای بنویسید که در آن دو ماتریس $n \times n$ را از ورودی گرفته و با استفاده از تابع predicate عناصر صفر آن‌ها را تبدیل به یک کرده و سپس دو ماتریس را در هم ضرب کند.

ب) کمک پردازنده برای اعداد مختلط: این کمک پردازنده باید قادر به انجام دستورات جمع، تفریق، ضرب، تقسیم، مقایسه، معکوس، مزدوج و تبدیل نمایش قطبی به نمایی و یا بالعکس روی اعداد مختلط باشد. همچنین، می‌توانید در صورت نیاز دستورالعمل‌های دسترسی به حافظه، کنترلی و پرشی برای کمک پردازنده در نظر بگیرید. کمک پردازنده باید بتواند یک عدد مختلط را به هر یک از دو فرمت قطبی و یا نمایی از حافظه خوانده و آن را در حافظه ذخیره نماید. پس از اتمام مراحل طراحی، کمک پردازنده‌ی خود را توسط برنامه‌ای که تمام دستورالعمل‌های نام برده در آن وجود داشته باشد، بیازمایید. پس از اطمینان از صحت اجرای تمامی دستورالعمل‌ها، با استفاده از دستورات اسمبلی برنامه‌ای بنویسید که در آن دو عدد مختلط به فرمت قطبی را از ورودی گرفته و مزدوج حاصل تقسیم را به فرمت نمایی ذخیره کند.

ج) کمک پردازنده برای اعداد ممیز شناور: این کمک پردازنده باید قادر به انجام دستورات جمع، تفریق، ضرب، تقسیم، مقایسه، معکوس و گرد کردن (به نزدیکترین عدد صحیح) روی اعداد ممیز شناور با دقت ساده[7] براساس استاندارد IEEE-754 باشد. همچنین، می‌توانید در صورت نیاز دستورالعمل‌های دسترسی به حافظه، کنترلی و پرشی برای کمک پردازنده در نظر

[6] Mask
[7] Single Precision

بگیرید. همانند موارد استثناء در نظر گرفته شده در استاندارد IEEE [8]، برای هر یک از موارد زیر باید در خروجی سیگنالی وجود داشته باشد که آن ها را گزارش کند. این موارد عبارتند از:

- Division by zero
- QNaN (quiet not a number)
- SNaN (signaling not a number)
- Inexact
- Underflow
- Overflow

پردازنده باید بتواند یک عدد ممیز شناور را از حافظه خوانده و آن را در حافظه ذخیره نماید. پس از اتمام مراحل طراحی، کمک پردازنده‌ی خود را توسط برنامه‌ای که تمام دستورالعمل‌های نام برده در آن وجود داشته باشد، بیازمایید. پس از اطمینان از صحت اجرای تمامی دستورالعمل‌ها، با استفاده از دستورات اسمبلی برنامه‌ای بنویسید که در آن دو عدد ممیز شناور را از ورودی گرفته، عدد بزرگتر را بر عدد کوچکتر تقسیم کرده و نتیجه‌ی گرد شده را ذخیره کند.

د) کمک پردازنده برای چند جمله‌ای‌ها: این کمک پردازنده باید قادر به انجام دستورات جمع، تفریق، ضرب، تقسیم، مشتق و محاسبه‌ی مقدار در یک نقطه روی چند جمله‌ای‌ها باشد. همچنین، می‌توانید در صورت نیاز دستورالعمل‌های دسترسی به حافظه، کنترلی و پرشی برای کمک پردازنده در نظر بگیرید. پردازنده باید بتواند یک چند جمله‌ای را از حافظه خوانده و آن را در حافظه ذخیره نماید. پس از اتمام مراحل طراحی، کمک پردازنده‌ی خود را توسط برنامه‌ای که تمام دستورالعمل‌های نام برده در آن وجود داشته باشد، بیازمایید. پس از اتمام مراحل طراحی، با استفاده از دستورات اسمبلی برنامه‌ای بنویسید که در آن یک چند جمله‌ای از ورودی گرفته و ریشه‌ی آن را با استفاده از روش نیوتن-رافسون محاسبه و ذخیره کند.

در صورت تشخیص تقلب، نمره صفر برای تقلب دهنده و گیرنده لحاظ خواهد شد.

این پروژه را در قالب گروه‌های ۴ نفره می‌توانید انجام دهید.

[8] IEEE Standard for Floating-Point Arithmetic," in *IEEE Std 754-2008* , vol., no., pp.1-70, 29 Aug. 2008, doi: 10.1109/IEEESTD.2008.4610935.