آزمایش شماره ۸: پیاده‌سازی یک کامپیوتر

هدف از این پروژه طراحی و پیاده‌سازی یک پردازنده ساده تک سیکل است که باید تمامی جزئیات مربوط به یک پردازنده (واحد کنترل، مسیر داده، شمارنده برنامه، Instruction Fetch، Decode و ....) با توجه به مطالب فراگرفته شده در درس معماری کامپیوتر برای آن در نظر گرفته شود.

کامپیوتر پایه دارای ویژگیهای زیر است:

- هشت عدد رجیستر ۸ بیتی به نام‌های R0 تا R7. این رجیستر بانک دارای دو پورت برای خواندن و یک پورت برای نوشتن است و خواندن از آن به صورت آسنکرون و نوشتن در آن به صورت سنکرون با کلاک و در لبه بالارونده کلاک انجام می‌شود. این رجیستر بانک باید به صورت یک ماژول مجزا در نظر گرفته شده و در ماژول اصلی از آن نمونه گرفته شود (پیاده‌سازی به شکل ساختاری).
- حافظه دستور (Instruction Memory که از این پس با نام IM شناخته می‌شود) به اندازه ۲۵۶خانه ۱۶بیتی. با توجه به مشخصات گفته شده، طول آدرس این حافظه و در نتیجه PC برنامه ۸ بیتی و همینطور دستورات و Instruction Register (IR)، ۱۶بیتی خواهند بود.
- حافظه داده (Data Memory که از این پس به آن DM گفته می‌شود) به اندازه ۲۵۶خانه ۸ بیتی.
- گذرگاه داده 8 (Data Bus) بیتی. در نتیجه عرض بیت همه‌ی داده‌ها و رجیسترها ۸ بیتی خواهد بود.
- کامپیوتر دارای ۴ پرچم (flag) است (CF-ZF-SF-OF) که این پرچم‌ها بعد از انجام دستورات خاصی بروزرسانی می‌شوند. نحوه‌ی بروزرسانی این پرچم‌ها در جدول ۱ توضیح داده شده است.
- کامپیوتر به صورت دو عملوندی است. این دو عملوند یا هر دو رجیستر هستند یا یک عملوند رجیستر و عملوند دیگر آدرس (آدرس DM یا IM) یا داده است. اگر بیت MSB دستور '۰' باشد، دو عملوند دستور از نوع رجیستر و یا یک عملوند رجیستر و عملوند دیگر یک داده ۳بیتی است. اگر بیت MSB دستور '۱' باشد، یک عملوند دستور از نوع رجیستر و عملوند دیگر یک داده و یا آدرس ۸بیتی است. (تشخیص نوع عملوند دوم از روی Opcode مشخص می‌شود)

دستورات کامپیوتر به همراه عملوندهای هر دستور، کد اسمبلی دستور و Flagهایی که به ازای هر دستور تحت تاثیر قرار می‌گیرند، در جدول زیر نشان داده شده است.

	Instruction	Operands	Description	Flags Affected	Assembly code
1	ADD	REG, REG	operand1 = operand1 + operand2	ALL Flags	0-000000001-RRRRRR
2	AND	REG, REG	Logical AND Result is stored in operand1	OF=0, CF=0 SF,ZF are updated	0-000000010-RRRRRR
3	SUB	REG, REG	operand1 = operand1 - operand2	ALL Flags	0-000000011-RRRRRR

4	OR	REG, REG	Logical OR Result is stored in operand1.	OF=0, CF=0 SF,ZF are updated	0-000000100-RRRRRR
5	XOR	REG, REG	Logical XOR Result is stored in operand1.	OF=0, CF=0 SF,ZF are updated	0-000000101-RRRRRR
6	MOV	REG, REG	Copy operand2 to operand1	NONE	0-000000110-RRRRRR
7	XCHG	REG, REG	Exchange values of two operands	NONE	0-000000111-RRRRRR
8	NOT	REG	One's complement negate operand1 = ~ operand1	NONE	0-000001000-RRRXXX
9	SAR	REG, imme-diate	Shift Arithmetic Right operand1. number of shifts set by operand2. the bit that goes off is set to CF. the sign bit that is inserted to the left-most position has the same value as before shift.	OF=0 if first operand keeps original sign. the bit that goes off is set to CF. SF,ZF are updated after shifting	0-000001001-RRRIII
10	SLR	REG , imme-diate	Shift Logical Right operand1. number of shifts set by operand2. the bit that goes off is set to CF. Zero bit is in-serted to the left-most position.	OF=0 if first operand keeps original sign. the bit that goes off is set to CF. SF,ZF are updated after shifting	0-000001010-RRRIII
11	SAL	REG , imme-diate	Shift Arithmetic Left operand1. number of shifts set by operand2. the bit that goes off is set to CF. Zero bit is inserted to the right most position.	OF=0 if first operand keeps original sign. the bit that goes off is set to CF. SF,ZF are updated after shifting	0-000001011-RRRIII
12	SLL	REG , imme-diate	Shift logical Left operand1. number of shifts set by operand2. the bit that goes off is set to CF. Zero bit is inserted to the right most position.	OF=0 if first operand keeps original sign. the bit that goes off is set to CF. SF,ZF are updated after shifting	0-000001100-RRRIII
13	ROL	REG , imme-diate	Rotate operand1 right. number of rotates set by operand2. shift all bits right, the bit that goes off is set to CF and the same bit is in-serted to the left-most position	OF=0 if first operand keeps original sign. the bit that goes off is set to CF. SF,ZF are updated after shifting	0-000001101-RRRIII

14	ROR	REG , immediate	Rotate operand1 left. number of rotates set by operand2. shift all bits left, the bit that goes off is set to CF and the same bit is inserted to the right-most position.	OF=0 if first operand keeps original sign. the bit that goes off is set to CF. SF,ZF are updated after shifting	0-000001110-RRRIII
15	INC	REG	operand = operand + 1	ALL (same as ADD)	0-000001111-RRRXXX
16	DEC	REG	operand = operand - 1	ALL (same as SUB)	0-000010000-RRRIII
17	NOP	NONE	NONE	NONE	0-000000000-XXXXXX
18	CMP	REG, REG	operand1 - operand2 Result is not stored anywhere, flags are set (OF, SF, ZF, CF) according to result.	ALL	0-000010100-RRRRRR
19	JE	Address of Memory	operand2 indicates the jump address of memory if jump happened operand1 is don't care Condition: ZF = 1	NONE	1-0000-XXXAAAAAAAA
20	JB	Address of Memory	operand2 indicates the jump address of memory if jump happened operand1 is don't care Condition: CF = 1 Used for unsigned compare	NONE	1-0001-XXXAAAAAAAA
21	JA	Address of Memory	operand2 indicates the jump address of memory if jump happened operand1 is don't care Condition: CF = 0, ZF = 0 Used for unsigned compare	NONE	1-0010-XXXAAAAAAAA
22	JL	Address of Memory	operand2 indicates the jump address of memory if jump happened operand1 is don't care Condition: SF not equal to OF Used for signed compare	NONE	1-0011-XXXAAAAAAAA
23	JG	Address of Memory	operand2 indicates the jump address of memory if jump happened operand1 is don't care Condition: SF = OF, ZF = 0 Used for unsigned compare	NONE	1-0100-XXXAAAAAAAA
24	JMP	Address of Memory	operand2 indicates the jump address of memory operand1 is don't care Jump without any condition	NONE	1-0101-XXXIIIIIII
25	LI	REG , immediate	Copy immediate value (operand2) to Register (operand1	NONE	1-1000-RRRAAAAAAAA
26	LM	REG, Address of Memory	Load data from data memory (address of memory in operand2) and copy to REG (operand1)	NONE	1-1001-RRRAAAAAAAA

27	SM	REG, Address of Memory	Store data from REG (operand1) to memory (address of memory in operand2)	NONE	1-1010-RRRAAAA AAAA

در پیاده‌سازی دستورات فوق نکات زیر را رعایت نمایید.

- مقدار immediate در دستورات ۹ تا ۱۴ حداکثر ۸ است. به همین دلیل قسمت immediate در این دستورات ۳بیتی است و همانند یک رجیستر فضا اشغال میکند. به همین دلیل این نوع دستورات را همانند REG-REG در نظر میگیریم. یعنی در این دستورات بیت MSB در کد '۰' است.
- از دستور ۱۹ دستورات نوع دوم (یکی از عملوندها ۳بیتی و دیگری ۸بیتی) شروع می‌شوند.
- در نمایش اسمبلی دستورات X به معنی Don't care، A به معنی آدرس (DM یا IM)، R به معنی آدرس یکی از رجیسترها و I به معنی مقدارضمنی می‌باشد. این مقدار در بعضی از دستورات ۳بیتی و در بعضی از دستورات دیگر ۸بیتی است.

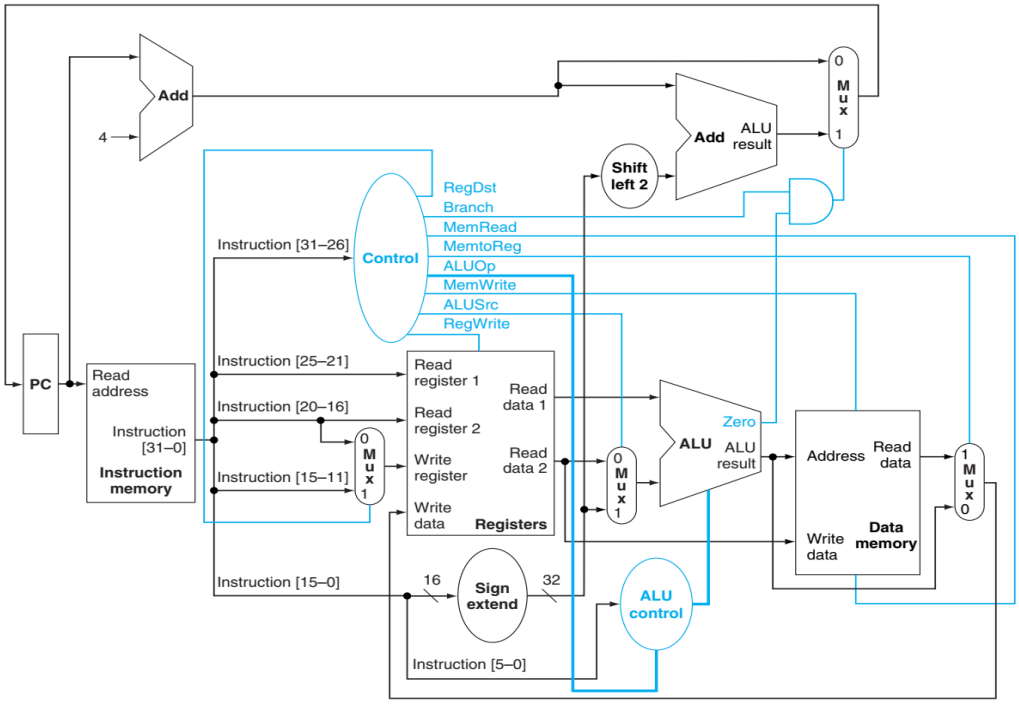
الف) طرح فوق را با استفاده از زبان توصیف سخت‌افزار (Verilog) پیاده‌سازی کنید.

ب) صحت عملکرد طرح خود را با استفاده از یک فایل آزمون مورد سنجش قرار دهید. فایل آزمون باید شامل برنامه‌ای برای انجام عمل جمع ۱۰ عدد روی پردازنده باشد. به این منظور نیاز است تا ابتدا یک برنامه به زبان اسمبلی و با استفاده از دستورات پردازنده بنویسید. سپس معادل باینری هر دستور و همچنین اعداد ورودی را در حافظه قرار دهید. از این پس با شروع شبیه‌سازی دستورات باید یک به یک وارد پردازنده شوند و پس از اتمام دستورات نتیجه عملیات روی ۵ ورودی در خانه شماره ۲۰۰ از حافظه ذخیره گردد (برنامه نوشته شده برای این عملیات باید شامل حلقه و با استفاده از دستورات پرش نوشته شود).

پیوست:

پردازنده تک سیکل

Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1



- ساختار کد:

```
//Controller
always @ ( posedge clk or ... )
begin
    .
    .
    .
end

//Datapath
always @ (c1, c2, ..)
begin
    .
    .
    .
end

//Logic Blocks Instantiation
ALU A1 (a, b, ...);
RegBank RB (a, b, ...);
InstMem IM (a, b, ...);
DataMem DM (a, b, ...);
```