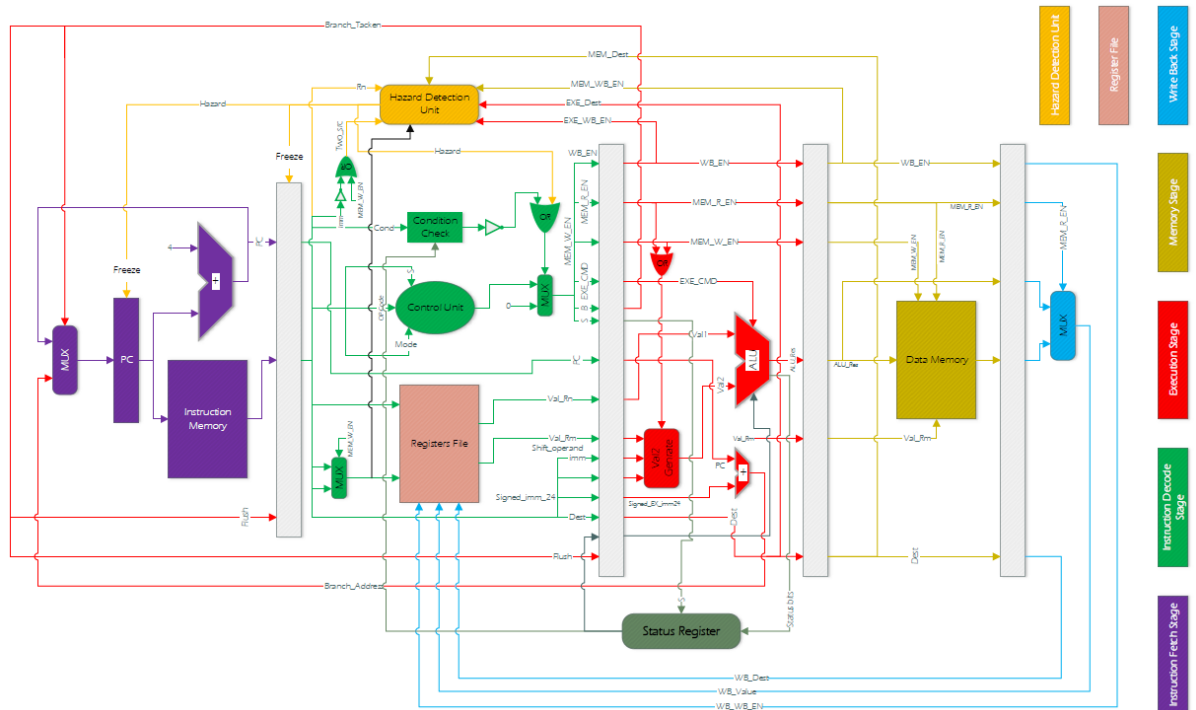


## توضیحات آزمایش

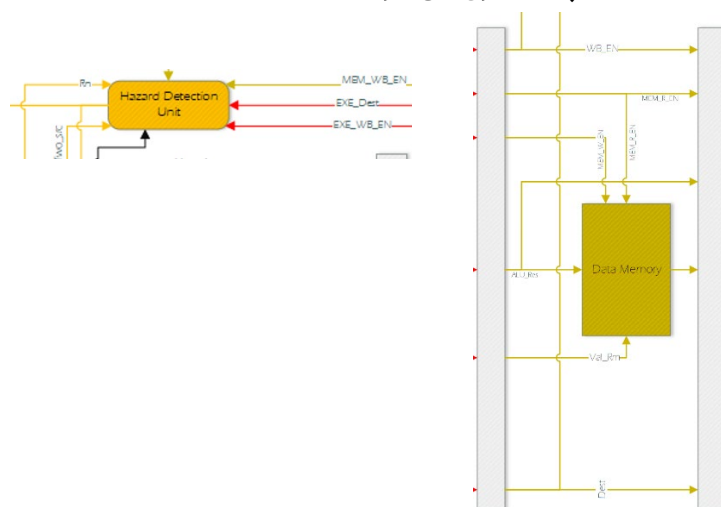
در این آزمایش پردازنده ARM به صورت پایپ‌لاین پیاده‌سازی می‌شود. دیاگرام این پردازنده به صورت زیر است:



این پردازنده دارای 13 دستور اصلی است. پیاده‌سازی باید در زبان ورپلاگ باشد و در نهایت پس از شبیه‌سازی در نرم‌افزار ModelSim، با استفاده از نرم‌افزار Quartus سنتز می‌شود و روی FPGA قرار می‌گیرد. سپس، با استفاده از یک تست‌بنچ، پردازنده پیاده‌سازی شده تست می‌شود. از اهداف این آزمایش می‌توان به یادگیری نحوه عیب‌یابی و تست مدارهای سخت‌افزاری طراحی شده اشاره کرد.

## جلسه چهارم

در این جلسه دو بخش مموری (MEM) و Hazard Unit پیاده‌سازی می‌شوند:



## ماژول MEM

این ماژول تنها شامل Data Memory است که یک ورودی 32 بیتی آدرس، یک ورودی 32 بیتی داده و بیت‌های memRead و memWrite را می‌گیرد و در صورت لزوم همگام با کلاک داده‌ای در مموری نوشته می‌شود و یا به صورت async از آن خوانده می‌شود.

در ابتدا آدرس ورودی منهای 1024 می‌شود چون که خانه 0 تا 1023 حافظه در پردازنده واقعی ARM، در اصل instruction memory است که در این آزمایش از حافظه جدا شده و در بخش IF قرار گرفته است.

سپس آدرس دو بیت به سمت راست شیفت می‌خورد. این برای ضرورت خواندن aligned از حافظه است (حافظه از خانه‌های 32 بیتی تشکیل شده است و در صورت نیاز به بایت شماره 2، بایت‌های 0، 1، 2، 3 خروجی داده می‌شوند).

کد این ماژول در ادامه آورده شده است:

```
module DataMemory(
    input clk, rst,
    input [31:0] memAdr, writeData,
    input memRead, memWrite,
    output reg [31:0] readData
);
    localparam WordCount = 64;

    reg [31:0] dataMem [0:WordCount-1]; // 256B memory

    wire [31:0] dataAdr, adr;
    assign dataAdr = memAdr - 32'd1024;
    assign adr = {2'b00, dataAdr[31:2]}; // Align address to the word boundary

    always @(negedge clk) begin
        if (memWrite)
            dataMem[adr] <= writeData;
    end

    always @(memRead or adr) begin
        if (memRead)
            readData = dataMem[adr];
    end
endmodule
```

لازم به ذکر است که شبیه‌سازی یک مموری 4 گیگ در ModelSim کار بسیار زمان‌بری خواهد بود و به همین دلیل از یک مموری 256 بایتی استفاده شده است.

همانطور که در دستور کار خواسته شده، مموری به صورت 64 خانه 4 بایتی پیاده‌سازی شده است.

## ماژول WB

این ماژول در جلسه قبلی پیاده‌سازی شده است.

## ماژول Hazard Detection Unit

این ماژول در حالات زیر data hazard را تشخیص می‌دهد و خروجی hazard را 1 می‌کند:

- wbEn دستوری که در مرحله Exe قرار دارد 1 باشد و Rn نیز برابر با destEx باشد.
- wbEn دستوری که در مرحله Exe قرار دارد 1 باشد، twoSrc فعال بوده و Rdm (ورودی دوم رجیسترفایل که Rm است به جز در دستور STR که Rd است) نیز برابر با destEx باشد.
- wbEn دستوری که در مرحله Mem قرار دارد 1 باشد و Rn نیز برابر با destMem باشد.

- wbEn دستوری که در مرحله Mem قرار دارد 1 باشد، twoSrc فعال بوده و Rdm نیز برابر با destMem باشد.

کد این ماژول در ادامه آورده شده است:

```
module HazardUnit(
    input [3:0] rn, rdm,
    input twoSrc,
    input [3:0] destEx, destMem,
    input wbEnEx, wbEnMem,
    output reg hazard
);
    always @(rn, rdm, destEx, destMem, wbEnEx, wbEnMem, twoSrc) begin
        hazard = 1'b0;
        if (wbEnEx) begin
            if (rn == destEx || (twoSrc && rdm == destEx)) begin
                hazard = 1'b1;
            end
        end
        if (wbEnMem) begin
            if (rn == destMem || (twoSrc && rdm == destMem)) begin
                hazard = 1'b1;
            end
        end
    end
endmodule
```

### ماژول TopLevel

تمامی موارد فوق در top level نیز افزوده شده و به هم متصل شده‌اند:

```
StageMem stMem(
    .clk(clk), .rst(rst),
    .wbEnIn(wbEnOutExMem), .memREnIn(memReadOutExMem), .memWEIn(memWriteOutExMem),
    .aluResIn(aluResOutExMem), .valRm(reg2OutExMem), .destIn(destOutExMem),
    .wbEnOut(wbEnOutMem), .memREnOut(memReadOutMem),
    .aluResOut(aluResOutMem), .memOut(memDataOutMem), .destOut(destOutMem)
);

RegsMemWb regsMem(
    .clk(clk), .rst(rst),
    .wbEnIn(wbEnOutMem), .memREnIn(memReadOutMem),
    .aluResIn(aluResOutMem), .memDataIn(memDataOutMem), .destIn(destOutMem),
    .wbEnOut(wbEnOutMemWb), .memREnOut(memReadOutMemWb),
    .aluResOut(aluResOutMemWb), .memDataOut(memDataOutMemWb), .destOut(destOutMemWb)
);

HazardUnit hzrd(
    .rn(hazardRn), .rdm(hazardRdm),
    .twoSrc(hazardTwoSrc),
    .destEx(destOutEx), .destMem(destOutMem),
    .wbEnEx(wbEnOutEx), .wbEnMem(wbEnOutMem),
    .hazard(hazard)
);
```

### بخش امتیازی

رجیسترفایل ساخته شده در این آزمایش از 15 رجیستر تشکیل شده است. این یعنی ورودی آدرس 4 بیت است که در صورتی که مقدار آن 4'd15 باشد، رجیسترفایل خروجی X می‌دهد.

مثال این حالت در دستور Branch به وجود می‌آید که 24 بیت سمت راست آن آدرس نسبی پرش است. از آنجا که بیت 0 تا 3 Rm و بیت 16 تا 19 Rn اند و پایپ‌لاین در هر صورت آنها را از register file می‌خواند، اگر آنها 1111b4 شوند خروجی رجیسترفایل X می‌شود. این در دستور Branch که کاری با مقدار رجیسترفایل ندارد تأثیری ندارد ولی اگر در یک دستور دیگر به اشتباه مقدار Rn یا Rm/Rd پانزده داده شود، می‌تواند مشکل‌زا باشد.

مثال دستور آخر برنامه محک که لوپ بینهایت است (چون که به آدرس PC+4-4 پرش می‌کند و روی خودش برمی‌گردد):

1110\_10\_1\_0\_111111111111111111111111 // B #-1 (PC = 32'd184)

/TopLevelTB/t/instOutfId	11101010...	111010101111...	00000...	111010101111...	00000...	111010101111...	00000...
/TopLevelTB/t/pcOutfId	188	188	192	0	188	192	0
/TopLevelTB/t/reg1OutfId	x			1024			1024
/TopLevelTB/t/reg2OutfId	x			1024			1024

همانطور که می‌بینیم، هر دو خروجی رجیسترفایل این دستور (که PC+4 = 188 مشخص شده است) X است. برای درست کردن این، با توجه به اینکه در پردازنده ARM واقعی، رجیستر شماره 15 همان PC است، در صورتی که ورودی رجیسترفایل 15 باشد، PC+4، در این مرحله وجود دارد را در نظر می‌گیریم. برای این کار از دو mux بعد از خروجی رجیسترفایل استفاده می‌کنیم و در صورت 15 بودن Rn یا Rdm (که معادل 1 بودن bitwise and ورودی است) خروجی را عوض می‌کنیم:

```
Mux2To1 #(32) muxRn15 (
    .a0 (regRn),
    .a1 (pcIn),
    .sel (&inst[19:16]),
    .out (reg1)
);

Mux2To1 #(32) muxRm15 (
    .a0 (regRm),
    .a1 (pcIn),
    .sel (&regfile2Inp),
    .out (reg2)
);
```

پس از اعمال تغییرات، خروجی به جای X، PC خواهد بود:

/TopLevelTB/t/instOutfId	11101010...	111010101111...	00000...	111010101111...	00000...	111010101111...	00000...
/TopLevelTB/t/pcOutfId	188	188	192	0	188	192	0
/TopLevelTB/t/reg1OutfId	188	188	192	1024	188	192	1024
/TopLevelTB/t/reg2OutfId	188	188	192	1024	188	192	1024

## تست پردازنده

تمامی دستورات محک در حافظه دستورات پردازنده قرار گرفته‌اند تا پردازنده به طور کامل تست شود. هدف نهایی این برنامه این است که عملیات Bubble Sort بر روی رجیسترهای R1 تا R4 صورت بپذیرد. دستورات را به ترتیب بررسی می‌کنیم:

1. MOV R0, #20 → R0 = 20

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	0	20
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	1	
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	2	
+ /TopLevelTB/t/stId/rf/regFile[3]	41	3	
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	4	

پس از این دستور مقدار R0 برابر با 20 شده است.

2. MOV R1, #4096 → R1 = 4096

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	20	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	1	4096
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	2	-1073741824
+ /TopLevelTB/t/stId/rf/regFile[3]	41	3	

پس از اجرای دستور مقدار R1 برابر با 4096 شده است.

3. MOV R2, #0xC0000000 → R2 = -1073741824

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	20	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4096	
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	2	-1073741824
+ /TopLevelTB/t/stId/rf/regFile[3]	41	3	
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	4	

مقدار R2 برابر با -1073741824 شده است.

4. ADDS R3, R2, R2 → R3 = -2147483648

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	20	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4096	
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824	
+ /TopLevelTB/t/stId/rf/regFile[3]	41	3	-2147483648
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	4	41
+ /TopLevelTB/t/stId/rf/regFile[5]	-123	5	

پس از اجرای این دستور نیز مقدار R3 برابر با -2147483648 شده است.

5. ADC R4, R0, R0 → R4 = 41

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	20	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4096	
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824	
+ /TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648	
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	4	41
+ /TopLevelTB/t/stId/rf/regFile[5]	-123	5	

مقدار R4 در این حالت برابر با 41 شده است.

6. SUB R5, R4, R4, LSL #2 → R5 = -123

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	20	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4096	
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824	
+ /TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648	
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	41	
+ /TopLevelTB/t/stId/rf/regFile[5]	-123	5	-123
+ /TopLevelTB/t/stId/rf/regFile[6]	10	6	

پس از اجرای دستور، مقدار R5 برابر با -123 شده است.



7. SBC R6, R0, R0, LSR #1 -> R6 = 10

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	20	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4096	
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824	
+ /TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648	
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	41	
+ /TopLevelTB/t/stId/rf/regFile[5]	-123	-123	
+ /TopLevelTB/t/stId/rf/regFile[6]	10	6	10
+ /TopLevelTB/t/stId/rf/regFile[7]	-123	7	

مقدار R6 برابر با 10 شده است.

8. ORR R7, R5, R2, ASR #2 -> R7 = -123

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	20	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4096	
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824	
+ /TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648	
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	41	
+ /TopLevelTB/t/stId/rf/regFile[5]	-123	-123	
+ /TopLevelTB/t/stId/rf/regFile[6]	10	10	
+ /TopLevelTB/t/stId/rf/regFile[7]	-123	7	-123
+ /TopLevelTB/t/stId/rf/regFile[8]	-2147483648	8	

مقدار R7 پس از اجرای دستور برابر با -123 شده است.

9. AND R8, R7, R3 -> R8 = -2147483648

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	20	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4096	
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824	
+ /TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648	
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	41	
+ /TopLevelTB/t/stId/rf/regFile[5]	-123	-123	
+ /TopLevelTB/t/stId/rf/regFile[6]	10	10	
+ /TopLevelTB/t/stId/rf/regFile[7]	-123	-123	
+ /TopLevelTB/t/stId/rf/regFile[8]	-2147483648	8	-2147483648
+ /TopLevelTB/t/stId/rf/regFile[9]	-11	9	-11

همانطور که در دستور خواسته شده، مقدار R9 برابر با -2146483648 شده است.

10. MVN R9, R6 -> R9 = -11

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	20	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4096	
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824	
+ /TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648	
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	41	
+ /TopLevelTB/t/stId/rf/regFile[5]	-123	-123	
+ /TopLevelTB/t/stId/rf/regFile[6]	10	10	
+ /TopLevelTB/t/stId/rf/regFile[7]	-123	-123	
+ /TopLevelTB/t/stId/rf/regFile[8]	-2147483648	-2147483648	
+ /TopLevelTB/t/stId/rf/regFile[9]	-11	9	-11

پس از اجرای این دستور نیز، مقدار R9 برابر با -11 شده است.

11. EOR R10, R4, R5 → R10 = -84

+	/TopLevelTB/tl/stId/rf/regFile[0]	1024	20	
+	/TopLevelTB/tl/stId/rf/regFile[1]	-2147483648	4096	
+	/TopLevelTB/tl/stId/rf/regFile[2]	-1073741824	-1073741824	
+	/TopLevelTB/tl/stId/rf/regFile[3]	41	-2147483648	
+	/TopLevelTB/tl/stId/rf/regFile[4]	8192	41	
+	/TopLevelTB/tl/stId/rf/regFile[5]	-123	-123	
+	/TopLevelTB/tl/stId/rf/regFile[6]	10	10	
+	/TopLevelTB/tl/stId/rf/regFile[7]	-123	-123	
+	/TopLevelTB/tl/stId/rf/regFile[8]	-2147483648	-2147483648	
+	/TopLevelTB/tl/stId/rf/regFile[9]	-11	-11	
+	/TopLevelTB/tl/stId/rf/regFile[10]	-1073741824	10	-84

مقدار R10 پس از اجرای این دستور برابر با -84 شده است.

12. CMP R8, R6 → z = 0

+	/TopLevelTB/tl/stId/rf/regFile[0]	1024	20	
+	/TopLevelTB/tl/stId/rf/regFile[1]	-2147483648	4096	8192
+	/TopLevelTB/tl/stId/rf/regFile[2]	-1073741824	-1073741824	
+	/TopLevelTB/tl/stId/rf/regFile[3]	41	-2147483648	
+	/TopLevelTB/tl/stId/rf/regFile[4]	8192	41	
+	/TopLevelTB/tl/stId/rf/regFile[5]	-123	-123	
+	/TopLevelTB/tl/stId/rf/regFile[6]	10	10	
+	/TopLevelTB/tl/stId/rf/regFile[7]	-123	-123	
+	/TopLevelTB/tl/stId/rf/regFile[8]	-2147483648	-2147483648	
+	/TopLevelTB/tl/stId/rf/regFile[9]	-11	-11	
+	/TopLevelTB/tl/stId/rf/regFile[10]	-1073741824	10	-84
+	/TopLevelTB/tl/stId/rf/regFile[11]	8192	11	
+	/TopLevelTB/tl/stMem/mem/dataMem[0]	-2147483648		
+	/TopLevelTB/tl/stMem/mem/dataMem[1]	-1073741824		
+	/TopLevelTB/tl/stMem/mem/dataMem[2]	41		
+	/TopLevelTB/tl/stMem/mem/dataMem[3]	8192		
+	/TopLevelTB/tl/stMem/mem/dataMem[4]	-123		
+	/TopLevelTB/tl/stMem/mem/dataMem[5]	10		
+	/TopLevelTB/tl/stMem/mem/dataMem[6]	-123		
+	/TopLevelTB/tl/stEx/statusRegister/out	0100	0001	1000

مقدار z (بیت دوم از سمت چپ)، برابر با 0 است.

13. ADDNE R1, R1, R1 → R1 = 8192

+	/TopLevelTB/tl/stId/rf/regFile[0]	1024	20	
+	/TopLevelTB/tl/stId/rf/regFile[1]	-2147483648	4096	8192
+	/TopLevelTB/tl/stId/rf/regFile[2]	-1073741824	-1073741824	
+	/TopLevelTB/tl/stId/rf/regFile[3]	41	-2147483648	
+	/TopLevelTB/tl/stId/rf/regFile[4]	8192	41	
+	/TopLevelTB/tl/stId/rf/regFile[5]	-123	-123	
+	/TopLevelTB/tl/stId/rf/regFile[6]	10	10	

مقدار R1 پس از اجرای دستور برابر با 8192 شده است.

14. TST R9, R8 → Z = 0

/TopLevelTB/tl/stId/rf/regFile[0]	1024	20		1024
/TopLevelTB/tl/stId/rf/regFile[1]	-2147483648	4096	8192	
/TopLevelTB/tl/stId/rf/regFile[2]	-1073741824	-1073741824		
/TopLevelTB/tl/stId/rf/regFile[3]	41	-2147483648		
/TopLevelTB/tl/stId/rf/regFile[4]	8192	41		
/TopLevelTB/tl/stId/rf/regFile[5]	-123	-123		
/TopLevelTB/tl/stId/rf/regFile[6]	10	10		
/TopLevelTB/tl/stId/rf/regFile[7]	-123	-123		
/TopLevelTB/tl/stId/rf/regFile[8]	-2147483648	-2147483648		
/TopLevelTB/tl/stId/rf/regFile[9]	-11	-11		
/TopLevelTB/tl/stId/rf/regFile[10]	-1073741824	-84		
/TopLevelTB/tl/stId/rf/regFile[11]	8192	11		
/TopLevelTB/tl/stMem/mem/dataMem[0]	-2147483648			
/TopLevelTB/tl/stMem/mem/dataMem[1]	-1073741824			
/TopLevelTB/tl/stMem/mem/dataMem[2]	41			
/TopLevelTB/tl/stMem/mem/dataMem[3]	8192			
/TopLevelTB/tl/stMem/mem/dataMem[4]	-123			
/TopLevelTB/tl/stMem/mem/dataMem[5]	10			
/TopLevelTB/tl/stMem/mem/dataMem[6]	-123			
/TopLevelTB/tl/stEx/statusRegister/out	0100	1000		

مقدار Z روی مقدار 0 باقی مانده است.

15. ADDEQ R2, R2, R2 → R2 = -1073741824

/TopLevelTB/tl/stId/rf/regFile[0]	1024	20		1024
/TopLevelTB/tl/stId/rf/regFile[1]	-2147483648	4096	8192	
/TopLevelTB/tl/stId/rf/regFile[2]	-1073741824	-1073741824		
/TopLevelTB/tl/stId/rf/regFile[3]	41	-2147483648		
/TopLevelTB/tl/stId/rf/regFile[4]	8192	41		
/TopLevelTB/tl/stId/rf/regFile[5]	-123	-123		
/TopLevelTB/tl/stId/rf/regFile[6]	10	10		
/TopLevelTB/tl/stId/rf/regFile[7]	-123	-123		

مقدار R2 تغییر نکرده و روی مقدار -1073741824 باقی مانده است. دلیل عدم تغییر این است که در عملیات قبلی بیت Z برابر با 0 بوده و شرط EQ برقرار نمی‌باشد.

16. MOV R0, #1024 → R0 = 1024

/TopLevelTB/tl/stId/rf/regFile[0]	1024	20		1024
/TopLevelTB/tl/stId/rf/regFile[1]	-2147483648	8192		
/TopLevelTB/tl/stId/rf/regFile[2]	-1073741824	-1073741824		
/TopLevelTB/tl/stId/rf/regFile[3]	41	-2147483648		
/TopLevelTB/tl/stId/rf/regFile[4]	8192	41		

مقدار R0 برابر با 1024 شده است.



17. STR R1, [R0], #0 -> MEM[1024] = 8192

+	/TopLevelTB/t/stId/rf/regFile[0]	1024	1024	
+	/TopLevelTB/t/stId/rf/regFile[1]	-2147483648	8192	
+	/TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824	
+	/TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648	
+	/TopLevelTB/t/stId/rf/regFile[4]	8192	41	
+	/TopLevelTB/t/stId/rf/regFile[5]	-123	-123	
+	/TopLevelTB/t/stId/rf/regFile[6]	10	10	
+	/TopLevelTB/t/stId/rf/regFile[7]	-123	-123	
+	/TopLevelTB/t/stId/rf/regFile[8]	-2147483648	-2147483648	
+	/TopLevelTB/t/stId/rf/regFile[9]	-11	-11	
+	/TopLevelTB/t/stId/rf/regFile[10]	-1073741824	-84	
+	/TopLevelTB/t/stId/rf/regFile[11]	8192	11	
+	/TopLevelTB/t/stMem/mem/dataMem[0]	-2147483648	8192	

مقدار Mem[0] (1024 - 1024) برابر با 8192 شده است.

18. LDR R11, [R0], #0 -> R11 = 8192

+	/TopLevelTB/t/stId/rf/regFile[0]	1024	1024	
+	/TopLevelTB/t/stId/rf/regFile[1]	-2147483648	8192	
+	/TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824	
+	/TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648	
+	/TopLevelTB/t/stId/rf/regFile[4]	8192	41	
+	/TopLevelTB/t/stId/rf/regFile[5]	-123	-123	
+	/TopLevelTB/t/stId/rf/regFile[6]	10	10	
+	/TopLevelTB/t/stId/rf/regFile[7]	-123	-123	
+	/TopLevelTB/t/stId/rf/regFile[8]	-2147483648	-2147483648	
+	/TopLevelTB/t/stId/rf/regFile[9]	-11	-11	
+	/TopLevelTB/t/stId/rf/regFile[10]	-1073741824	-84	
+	/TopLevelTB/t/stId/rf/regFile[11]	8192	11	8192
+	/TopLevelTB/t/stMem/mem/dataMem[0]	-2147483648	8192	

مقدار R11 برابر با 8192 شده است.

19. STR R2, [R0], #4 -> MEM[1028] = -1073741824

+	/TopLevelTB/t/stId/rf/regFile[0]	1024	1024	
+	/TopLevelTB/t/stId/rf/regFile[1]	-2147483648	8192	
+	/TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824	
+	/TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648	
+	/TopLevelTB/t/stId/rf/regFile[4]	8192	41	
+	/TopLevelTB/t/stId/rf/regFile[5]	-123	-123	
+	/TopLevelTB/t/stId/rf/regFile[6]	10	10	
+	/TopLevelTB/t/stId/rf/regFile[7]	-123	-123	
+	/TopLevelTB/t/stId/rf/regFile[8]	-2147483648	-2147483648	
+	/TopLevelTB/t/stId/rf/regFile[9]	-11	-11	
+	/TopLevelTB/t/stId/rf/regFile[10]	-1073741824	-84	
+	/TopLevelTB/t/stId/rf/regFile[11]	8192	11	8192
+	/TopLevelTB/t/stMem/mem/dataMem[0]	-2147483648	8192	
+	/TopLevelTB/t/stMem/mem/dataMem[1]	-1073741824	-1073741824	

مقدار Mem[1] برابر با -1073741824 شده است.

20. STR R3, [R0], #8 -> MEM[1032] = -2147483648

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	1024
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	8192
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824
+ /TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	41
+ /TopLevelTB/t/stId/rf/regFile[5]	-123	-123
+ /TopLevelTB/t/stId/rf/regFile[6]	10	10
+ /TopLevelTB/t/stId/rf/regFile[7]	-123	-123
+ /TopLevelTB/t/stId/rf/regFile[8]	-2147483648	-2147483648
+ /TopLevelTB/t/stId/rf/regFile[9]	-11	-11
+ /TopLevelTB/t/stId/rf/regFile[10]	-1073741824	-84
+ /TopLevelTB/t/stId/rf/regFile[11]	8192	8192
+ /TopLevelTB/t/stMem/mem/dataMem[0]	-2147483648	8192
+ /TopLevelTB/t/stMem/mem/dataMem[1]	-1073741824	-1073741824
+ /TopLevelTB/t/stMem/mem/dataMem[2]	41	-2147483648

مقدار Mem[2] برابر با -2147483648 شده است.

21. STR R4, [R0], #13 -> MEM[1036] = 41

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	1024
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	8192
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824
+ /TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	41
+ /TopLevelTB/t/stId/rf/regFile[5]	-123	-123
+ /TopLevelTB/t/stId/rf/regFile[6]	10	10
+ /TopLevelTB/t/stId/rf/regFile[7]	-123	-123
+ /TopLevelTB/t/stId/rf/regFile[8]	-2147483648	-2147483648
+ /TopLevelTB/t/stId/rf/regFile[9]	-11	-11
+ /TopLevelTB/t/stId/rf/regFile[10]	-1073741824	-84
+ /TopLevelTB/t/stId/rf/regFile[11]	8192	8192
+ /TopLevelTB/t/stMem/mem/dataMem[0]	-2147483648	8192
+ /TopLevelTB/t/stMem/mem/dataMem[1]	-1073741824	-1073741824
+ /TopLevelTB/t/stMem/mem/dataMem[2]	41	-2147483648
+ /TopLevelTB/t/stMem/mem/dataMem[3]	8192	41

پس از اجرای دستور، مقدار Mem[3] برابر با 41 شده است.

22. STR R5, [R0], #16 -> MEM[1040] = -123

+ /TopLevelTB/t/stMem/mem/dataMem[0]	-2147483648	8192
+ /TopLevelTB/t/stMem/mem/dataMem[1]	-1073741824	-1073741824
+ /TopLevelTB/t/stMem/mem/dataMem[2]	41	-2147483648
+ /TopLevelTB/t/stMem/mem/dataMem[3]	8192	41
+ /TopLevelTB/t/stMem/mem/dataMem[4]	-123	-123

مقدار Mem[4] برابر با -123 شده است.

23. STR R6, [R0], #20 -> MEM[1044] = 10

+ /TopLevelTB/t/stMem/mem/dataMem[0]	-2147483648	8192
+ /TopLevelTB/t/stMem/mem/dataMem[1]	-1073741824	-1073741824
+ /TopLevelTB/t/stMem/mem/dataMem[2]	41	-2147483648
+ /TopLevelTB/t/stMem/mem/dataMem[3]	8192	41
+ /TopLevelTB/t/stMem/mem/dataMem[4]	-123	-123
+ /TopLevelTB/t/stMem/mem/dataMem[5]	10	10



همانطور که مشاهده می‌شود، مقدار [5]Mem برابر با 10 شده است.

24. LDR R10, [R0], #4 -> R10 = -1073741824

+ /TopLevelTB/t/stId/rf/regFile[6]	10	10	
+ /TopLevelTB/t/stId/rf/regFile[7]	-123	-123	
+ /TopLevelTB/t/stId/rf/regFile[8]	-2147483648	-2147483648	
+ /TopLevelTB/t/stId/rf/regFile[9]	-11	-11	
+ /TopLevelTB/t/stId/rf/regFile[10]	-1073741824	-84	-1073741824
+ /TopLevelTB/t/stId/rf/regFile[11]	8192	8192	

مقدار R10 برابر با -1073741824 شده است.

25. STR R7, [R0], #24 -> MEM[1048] = -123

+ /TopLevelTB/t/stMem/mem/dataMem[0]	-2147483648	8192	
+ /TopLevelTB/t/stMem/mem/dataMem[1]	-1073741824	-1073741824	
+ /TopLevelTB/t/stMem/mem/dataMem[2]	41	-2147483648	
+ /TopLevelTB/t/stMem/mem/dataMem[3]	8192	41	
+ /TopLevelTB/t/stMem/mem/dataMem[4]	-123	-123	
+ /TopLevelTB/t/stMem/mem/dataMem[5]	10	10	
+ /TopLevelTB/t/stMem/mem/dataMem[6]	-123	-123	

پس از اجرای دستور، مقدار [6]Mem برابر با -123 شده است.

26. MOV R1, #4 -> R1 = 4

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	1024	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	8192	4
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741824	
+ /TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648	

مقدار R1 برابر با 4 شده است.

27. MOV R2, #0 -> R2 = 0

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	1024	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	8192	4
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	-1073741...	0
+ /TopLevelTB/t/stId/rf/regFile[3]	41	-2147483648	

پس از اجرای این دستور، مقدار R2 برابر با 0 می‌شود.

28. MOV R3, #0 -> R3 = 0

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	1024	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4	
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	0	
+ /TopLevelTB/t/stId/rf/regFile[3]	41	-214...	0
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	41	

پس از اجرای این دستور نیز، مقدار R3 برابر با 0 می‌شود.

29. ADD R4, R0, R3, LSL #2 -> R4 = 1024

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	1024	
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4	
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	0	
+ /TopLevelTB/t/stId/rf/regFile[3]	41	0	
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	41	1024

مقدار R4 برابر با 1024 می‌شود.

30. LDR R5, [R4], #0 → R5 = 8192

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	1024		
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4		
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	0		
+ /TopLevelTB/t/stId/rf/regFile[3]	41	0		
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	1024		
+ /TopLevelTB/t/stId/rf/regFile[5]	-123	-123	8192	

همانطور که مشاهده می‌شود، مقدار R5 برابر با 8192 شده است.

31. LDR R6, [R4], #4 → R6 = -1073741824

+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	0		
+ /TopLevelTB/t/stId/rf/regFile[3]	41	0		
+ /TopLevelTB/t/stId/rf/regFile[4]	8192	1024		
+ /TopLevelTB/t/stId/rf/regFile[5]	-123	8192		
+ /TopLevelTB/t/stId/rf/regFile[6]	10	10	-1073741824	

در این دستور نیز مقدار R6 برابر با -1073741824 می‌شود.

32. CMP R5, R6 → z = 0, n = 0, v = 0

+ /TopLevelTB/t/stMem/mem/dataMem[4]	-123	-123		
+ /TopLevelTB/t/stMem/mem/dataMem[5]	10	10		
+ /TopLevelTB/t/stMem/mem/dataMem[6]	-123	-123		
+ /TopLevelTB/t/stEx/statusRegister/out	0100	1...0010		

مقدار z, n, v و برابر با 0 شده است و فقط مقدار c برابر با 1 شده است.

33. STRGT R6, [R4], #0 → MEM[1024] = -1073741824

+ /TopLevelTB/t/stId/rf/regFile[8]	-2147483648	-2147483648		
+ /TopLevelTB/t/stId/rf/regFile[9]	-11	-11		
+ /TopLevelTB/t/stId/rf/regFile[10]	-1073741824	-1073741824		
+ /TopLevelTB/t/stId/rf/regFile[11]	8192	8192		
+ /TopLevelTB/t/stMem/mem/dataMem[0]	-2147483648	8192	-1073741824	

در این بخش نیز مقدار Mem[0] برابر با -1073741824 است.

34. STRGT R5, [R4], #4 → MEM[1028] = 8192

+ /TopLevelTB/t/stId/rf/regFile[11]	8192	8192		
+ /TopLevelTB/t/stMem/mem/dataMem[0]	-2147483648	-1073741824		
+ /TopLevelTB/t/stMem/mem/dataMem[1]	-1073741824	8192		
+ /TopLevelTB/t/stMem/mem/dataMem[2]	41	-2147483648		

در این بخش نیز Mem[1] برابر با 8192 شده است.

35. ADD R3, R3, #1 → R3 = 1

+ /TopLevelTB/t/stId/rf/regFile[0]	1024	1024		
+ /TopLevelTB/t/stId/rf/regFile[1]	-2147483648	4		
+ /TopLevelTB/t/stId/rf/regFile[2]	-1073741824	0		
+ /TopLevelTB/t/stId/rf/regFile[3]	41	0		1

پس از انجام این دستور، مقدار R3 برابر با 1 شده است.

36. CMP R3, #3 → z = 0, n = 1, v = 0

+ /TopLevelTB/t/stMem/mem/dataMem[4]	-123	-123		
+ /TopLevelTB/t/stMem/mem/dataMem[5]	10	10		
+ /TopLevelTB/t/stMem/mem/dataMem[6]	-123	-123		
+ /TopLevelTB/t/stEx/statusRegister/out	0100	0010	1010	

در این دستور، مقادیر n و c برابر با 1 شده‌اند.



37. BLT #-9 -> PC = 32'd112

/TopLevelTB/tl/stMem/mem/dataMem[6]	-123	-123					
/TopLevelTB/tl/stEx/statusRegister/out	1010	1010					
/TopLevelTB/tl/stIf/pcReg/out	120	148	152	112	116		

پس از اجرای این دستور، مقدار PC برابر با 112 شده است.

با توجه به اینکه باقی دستورات در لوپ قرار دارند، جهت جلوگیری از تکرار بی دلیل، از گذاشتن تصویر نتایج جلوگیری شده است.

همانطور که پیش‌تر ذکر شد، هدف این برنامه، مرتب‌سازی رجیسترهای R1 تا R4 به ترتیب صعودی است. تصویر خروجی این رجیسترها پس از پایان برنامه به شرح زیر است:

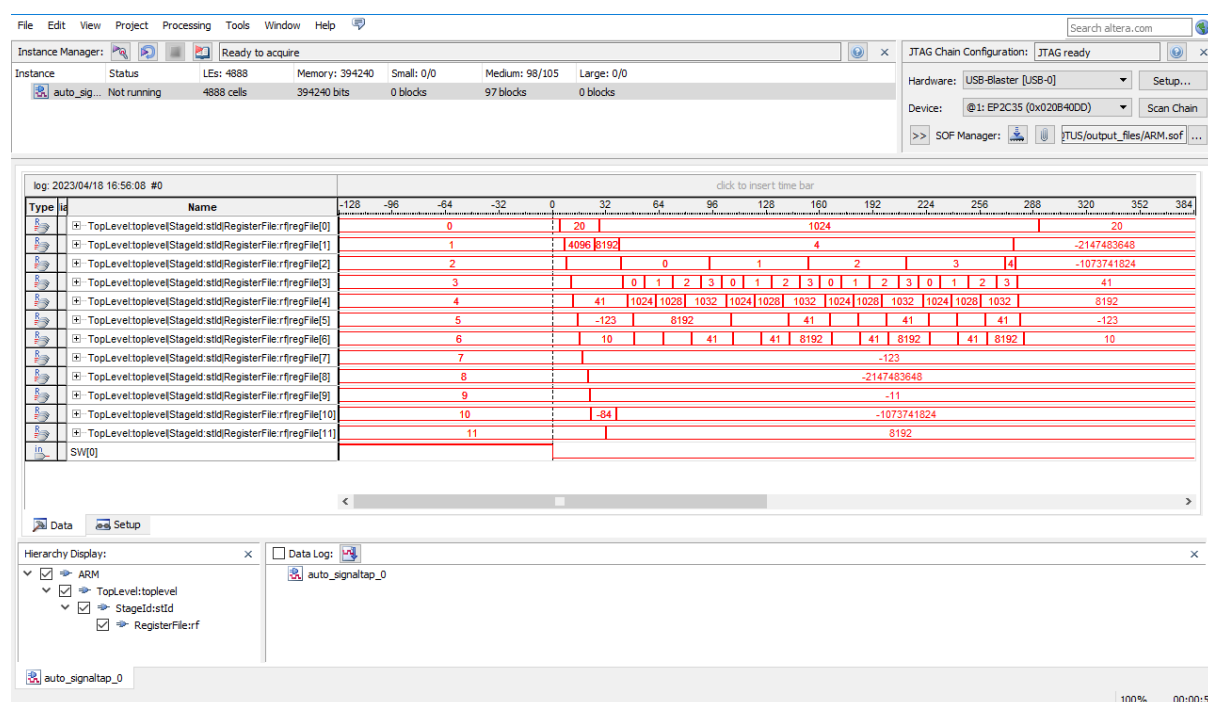
/TopLevelTB/tl/std/rf/regFile[0]	1024	1024					
/TopLevelTB/tl/std/rf/regFile[1]	-2147483648	4	-2147483648				
/TopLevelTB/tl/std/rf/regFile[2]	-1073741824	4	-1073741824				
/TopLevelTB/tl/std/rf/regFile[3]	41	3		41			
/TopLevelTB/tl/std/rf/regFile[4]	8192	1032			8192		
/TopLevelTB/tl/std/rf/regFile[5]	-123	41			-123		
/TopLevelTB/tl/std/rf/regFile[6]	10	8192					10
/TopLevelTB/tl/std/rf/regFile[7]	-123	-123					
/TopLevelTB/tl/std/rf/regFile[8]	-2147483648	-2147483648					
/TopLevelTB/tl/std/rf/regFile[9]	-11	-11					
/TopLevelTB/tl/std/rf/regFile[10]	-1073741824	-1073741824					
/TopLevelTB/tl/std/rf/regFile[11]	8192	8192					

همانطور که مشاهده می‌شود، این رجیسترها به صورت صعودی مرتب شده‌اند.

## نتایج سنتز

Flow Summary	
Flow Status	Successful - Thu Apr 20 23:10:01 2023
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	ARM
Top-level Entity Name	ARM
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	7,753 / 33,216 ( 23 % )
Total combinational functions	4,066 / 33,216 ( 12 % )
Dedicated logic registers	5,853 / 33,216 ( 18 % )
Total registers	5853
Total pins	418 / 475 ( 88 % )
Total virtual pins	0
Total memory bits	396,288 / 483,840 ( 82 % )
Embedded Multiplier 9-bit elements	0 / 70 ( 0 % )
Total PLLs	0 / 4 ( 0 % )

## خروجی برنامه محک در SignalTap



همانطور که مشاهده می‌شود، پس از اتمام برنامه، مقادیر به صورت مرتب شده در رجیسترهای R1 تا R4 قرار گرفته‌اند.