

MTH410E – RISC-V Architecture and Processor Design

Homework 4 – SuperScalar RISC-V Processor

Write a SystemVerilog code of a pipelined/2-way superscalar RISC-V processor. The designed micro-architecture should include all the instructions in RV32I. The processor should work as in-order and single-issue, so there is no need for FENCE instruction implementation. The processor should resolve the all data control hazards either by stalling or full bypassing/scoreboard (+20 bonus points).

The datapath of the processor should be as follows;

Datapath A (1 cycle): ALU operations (RV32I except memory operations)
Datapath B (1 cycle): MEM + ALU operations (RV32I)

The top file of the processor should be as follows;

```
module riscv_superscalar (
    parameter DMemInitFile = "dmem.mem"; // data memory initialization file
    parameter IMemInitFile = "imem.mem"; // instruction memory initialization file
    parameter LogFile = "rv_track.txt"; // log file to track the processor state and used for verification/grading
)
(
    input clk_i; // input clock
    input rst_ni; // logic-0 asserted asynch reset
    output [31:0] rf_data_hex // just for synthesis, otherwise the tools can remove everything for optimization
);

// module body
// you can also use multiple modules in different files

endmodule
```

The data and instruction memories of the processor should be initialized by *DMemInitFile* and *IMemInitFile* respectively (i.e., use \$readmemh in the initial block of the memories).

To trace the processor state, RF write and DMEM write operations should be logged to *LogFile*. Therefore, following two functions should be used in RF and DMEM modules to track the written values. You can change the signal names (e.g., rf_idx_dec) with respect to your signal names.

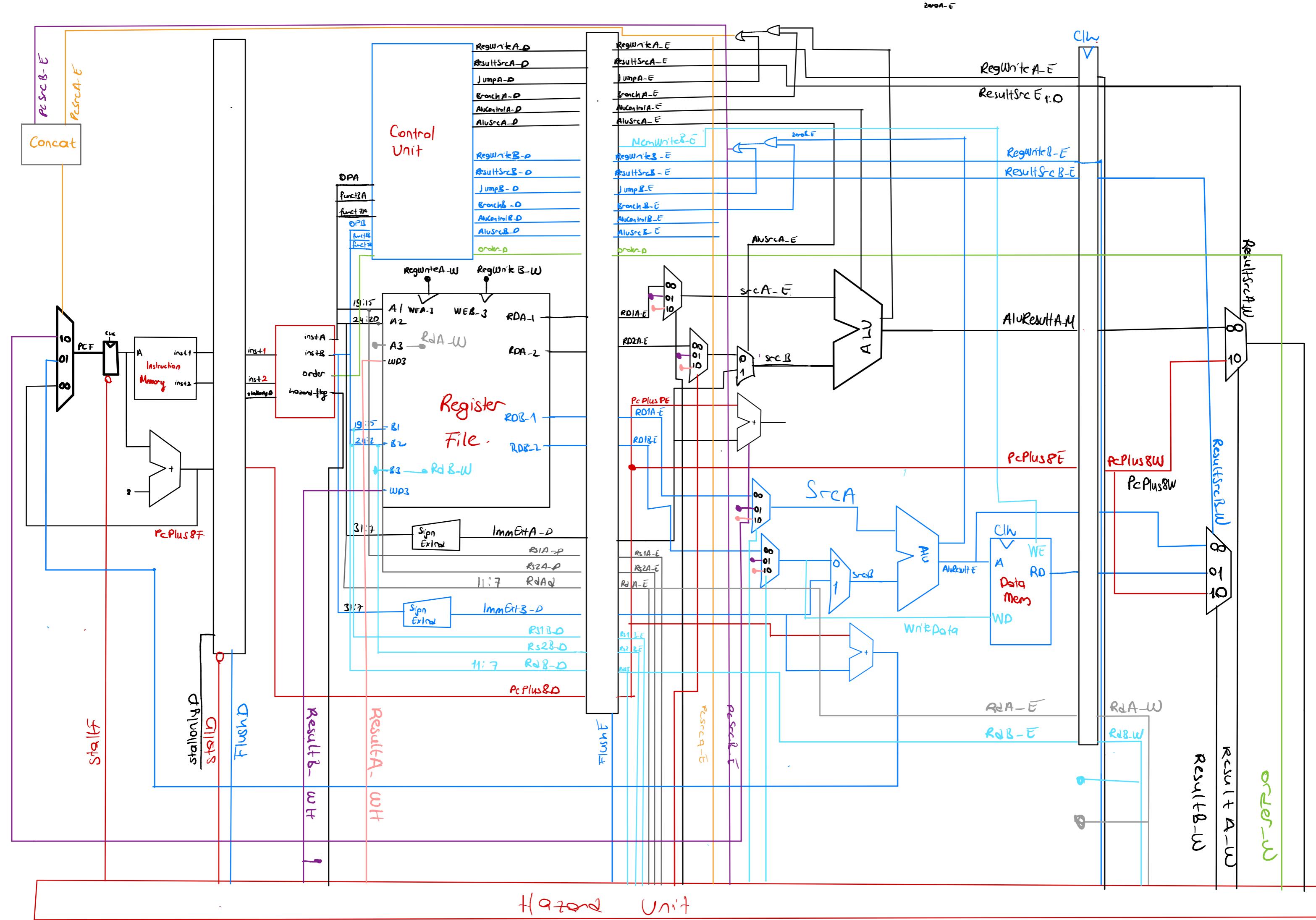
```
$fwrite(LogFile, "x%0d 0x%16h", rf_idx_dec, rf_data_hex); // log the register file writes
$fwrite(LogFile, "mem 0x%h 0x%h", dmem_idx_dec, dmem_data_hex); // log the data memory writes
```

Deadline: 26/12/2023 23:59

Some resources;

- [RISC-V: An Overview of the Instruction Set Architecture](#)

Ödevi yaparken gerekli olan uprogram, anfı test yapılmaya seviyem.
Bu rapor da öneridipin mindiyi anlatır.



Risc-V Superscalar Final Ödevi

Ödevde birde 2 yolu superscalar Risc-V processor tasarılamamız isteniyor. Ben ayrıca bypassing yapmaya de çalıştım.

Ayrıca ödevde 4 stagede bu işe sunulmamız bekleniyor.

- Fetch
- Decode
- Execute
- Write stages.

Superscalar yapida iki instruction yaparken aynı zamanda pipeline de olması bekleniyor.
Bunisce birkaç instruction deneyelim;

add x3, x4, x5	F	D	E	W
or x6, x1, x2	F	D	E	W
lw x7, x0(5)	F	D	E	W
not x9, x10, x11	F	D	E	W

Bu con optimistik instructionler oldu açısından, malesef birim ugrasmamız gereken hazardlar
Bu hazardlar 2-ödevde de karşımıza sıkışır
processorun pipelined yapısından gelen hazardlar,
ve bu ödevde deredipimiz 2-way superscalar
yapıdan gelen hazardlar. Bunları daha detaylı inceleyelim:

1.) Prosesörün Superscalar Yapısından Gelen Hazzardlar

a.) 2 load ya da save instruction'u gelirse;

Sadece B pathinde Mem bloğumuz var dolayısıyla aynı anda iki lw, sw instruction'u alamayız. Bu durumda önce ilk geleni sonra ikinci geleni almamız gerektir. Ve bu durumda pesi sona gelecek instructionları da durduramamız gerektir. HAZARD

Case

lw $x_3, \overline{s}(x_0)$

F D E W

sw $x_9, b(x_1)$

F D D E W

add x_4, x_5, x_7

F F D E W

or x_6, x_1, x_2

F F D E W

} bu forms
gelişilmeli
Hazzardı
 önlemek için

b.) RAW durumu (Read After Write)

Aynı anda çalıştığımız iki instruction'da 1. instruction'un destination registeri (rd) eğer 2. instruction'un source registerindeki herhangi birine erit olursa bunda ilkinin sonucunu almamız gerekebilir dolayısıyla 1 cycle beklemeliyiz. HAZARD!

Case

add x_3, x_2, x_1 F D E W

or x_4, x_3, x_5 F D D E W

lw $x_8, \overline{s}(x_0)$ F F D E W

not x_6, x_7, \overline{s} F F D E W

} ilki gibi ikinci instruction'u 1 cycle beklettikten ve Fetch'i bir cycle durdurdu.

c.) 1. instruction'un beg ya da gel olması

Per pese iki instruction geldiginde illü beg instructioni ise ikincinin execute kumune girmeyip decode'la kalması gerekiyor.
Alu'dan silen sonucu şöre devam edebilir ya da FlushE aktifleşerek silinir. ✓

FlushE, ve FlushD kullanmamız yeterli.

beg x4, x5, 24 FD E W
sub x8, x1, x3 FD D E W
or x3, x6, x5 FD E W
not x11, x10, x12. FD E W
;
;
;

L4: add x7, x3, x1

Yanlış İc
olması
seçenek

olması gereken

beg x4, x5, 24 FD E W
sub x8, x1, x3 FD ---
or x3, x6, x5 F ---
not x11, x10, x12. F ---
;
;
;

Dogrular
ise

L4: add x7, x3, x1 F

ikinci instruction beg sjal ise sonu yok.

Yukarıdaki 3 case'de aynı şekilde çözülebilir:

3 case'den en az biri (herhangi bir-) tespit edildikten sonra çözümü zu şekilde olmalı;

- 1.) Decode aşamasında tespit edildikten sonra hazard-flag sinyali aktifleştir.
- 2.) Hazard-flag sinyali hazard-unit'e girer.
- 3.) Hazard-flag sinyali aktif ise instructionlarda iki 3 pathine girdi, 2-inst. bekler.
- 4.a) Hazard unit'te hazard flag kontrol edilir aktif ise stallonly sinyali aktifleştir.
- 4.b) 1 cycle sonra stallonly sinyali aktif ise 2-instruction 3 pathine gider.
- 4.c) Aynı şekilde stallF sinyali de aktifleştir ve F stage' i de durdurur.

F D E W

F D D E W \rightarrow stallonly aktif

F F D E W

F F D E W

stallF aktif

d.) Destination hazard.

Aynı anda olasıpmış iki instruction'un her ikisinin de destination register aynı ise 2-gelen in yazdırılması geçer. Bu da hazard unit ile seq/leaf/cap'ız.

4. Özüm:

gelen instrüksiyonun order'i tut.

order 00 \rightarrow 1 \rightarrow A
2 \rightarrow B pathinde

order 01 \rightarrow 1 instruction von B pathinde

order 10 \rightarrow 1 \rightarrow B pathinde
2 \rightarrow A

Pseudocode

if $R_{RAW} == R_{WB}$:

if (order = 10)

Result A-WH = ResultA-W;

Result B-WH = ResultA-W;

else

ResultA-WH = ResultB-W;

ResultB-WH = ResultB-W;

01 ise A
ve B rd'ler
erit olmuyor
zaten.

Odevin ilk basinda 3 case oldugunu

devindigini isin order'i 2 bit verdin.

ove 1 logigi ile de maledilebilirmiz.

2.) Pipeline Yapisidon Gelen Hazarlar,

a.) RAW Hazardı

Zaten 4 stageimiz var, RAW
durumda bir onceki instruksiyon rd

degeri bir sonrakinin rs'line erit
oldugunu devamde ama per pesse din-
meyen instruksiyonlar isin de gecerli

bunun için Forward A ve Forward B

sinyalleri ile hazardı önlüyor.

hazardı

önlüyor.

Örnek case

add x1, x2, x3

F D E W

or x4; x5, x6

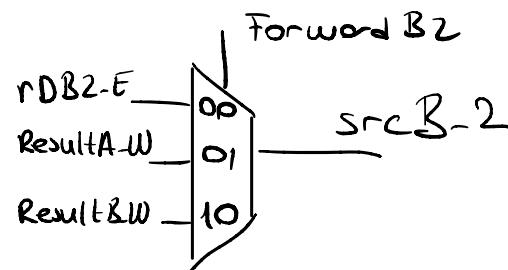
F D E W

not x7, x8, x9

F D E W

xor x10, x11, x4

F D E W



toplam 4 tane Forward sinyali ile select eden muxumuz var ve Forward sinyollarını 3 case' i sikiştererek 2 bit ile represente edebiliyor

00: rd**-E (rd1A-E, rd1B-E, rd2A-E, rd2B-E)

01: ResultA-W

10: ResultB-W

b.) beq inceleme

Aynı anda gelen beq': incededik ikinci instructionda geçtiğinde sayt ve PC atlarsa

addi x11, x12, x13

F D E M W

or x14, x6, 7

F D E M W

add x1, x2, x3

F D E W

beq x4, 5, 24

F D E ---

or x7, x8, x9

F D } FlushE

lw x10, 5(x0)

F D }

add x21, x22, 5

F D } FlushD

or x23, x24, b

F }

24: not x15, x16, x17

F

and x18, x19, x20

F

Bu durumlar göz önüne alarak hazard-unit'i inceleyelim:

① Hazard-flag sinyali

→ Aktifler → stallF, stallonly sinyalleini aktif

(1) et.

② PcsrcA-E ya da Pcsrb-E aktifse;

1. beq ise → B path

2. beq ise → A path (1.'si beq degil)

→ FlushD ve FlushE aktif et

Her ikisi aynı anda aktif olamaz sistem gereği.

③ RegWriteA-W aktifler, RstAE ya da Rst2AE ya da RstBE ya da Rst2BE registerlerin herhangi biri R2AW'ye eritse; YA DA

RegWriteB-W aktifler, RstAE ya da
Rst2AE ya da RstBE ya da Rst2BE
registerlerin herhangi biri R2BW'ye
eritse; burada mudahale etmemiz gerekken
bir durum vardır.

RegWriteA-W ve RegWriteB-W her ikisi
de eritse order'a bakularak ResultA-WH
ve ResultB-WH güncellenir (yukonda asildedi)

buna gör Pseudocode RS1AE, RS1BE, RS2AE, RS2BE

if (RegWrite A-WH & & $(RS^{**}E == Rd AW)$)
 Forward $E = 01$

if (RegWrite B-WH & & $(RS^{**}E == Rd AW)$)
 Forward $E = 10$ olurdu düzeltilecektir

Örnek Instruction

addi , x1, 5 - 19
 addi , x2, 6 - 19
 addi , x3, 7 - 19
 addi , x4, 8 - 19
 add , x5, x1, x2 - 51
 add , x9, x1, x3 - 51
 sw x3, 5(x0) 35
 lw x6, 5(x0) 3

