COMPENG 4DM4 Assignment 1 Report

Aaron Pinto
pintoa9

Raeed Hassan
hassam41

October 24, 2022

Exercise Part (A) - The 7-stage RISC Pipeline

(A1) There is a 2 clock-cycle stall on LOAD instructions when the instruction following a LOAD instruction uses the value being loaded. The timing diagram can be seen in Figure 1.

(A2) The number of branch-delay-slots following a BRANCH instruction will be 2 clock cycles in both cases considered:

1. If the value R0 being tested is available in a register at the start of the ID stage, then the "branch is resolved" in the ID stage - the ID stage will compare R0 with zero, and generate a "Take-Branch" signal which is sent back to the IF1 stage from the ID stage in clock-cycle 3. In this case, 2 branch-delay-slots will occur, as seen in Figure 2. The compiler can attempt to fill the "BRANCH+1" and "BRANCH+2" branch delay slots with useful instructions, otherwise fill with a NO-OP.

2. If the value R0 is being computed in the EX stage, by the previous instruction, and R0 needs to be forwarded to the ID stage, then we do not "stretch the clock" to allow R0 to be tested in the ID stage in the same clock cycle. The system will "stall" the BNEZ for 1 extra clock cycle, waiting for R0 to be computed and forwarded to the ID stage. In this case, 2 branch-delay-slots will occur, as seen in Figure 3. The compiler can attempt to fill the "BRANCH+1" and "BRANCH+2" branch delay slots with useful instructions, otherwise fill with a NO-OP.

	Clock Cycle														
Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LW R0,0(R1)	IF1	IF2	ID	EX	MEM1	MEM2*	WB				*forward R0 (MEM2* to *EX) in cc6				
ADD R3,R0,R2		IF1	IF2	ID	stall	stall	*EX	MEM1	MEM2	WB					
LOAD+2			IF1	IF2	stall	stall	ID	EX	MEM1	MEM2	WB				
LOAD+3				IF1	stall	stall	IF2	ID	EX	MEM1	MEM2				
LOAD+4					stall	stall	IF1	IF2	ID	EX	MEM1	MEM2			

Figure 1: Timing Diagram for A1

	Clock Cycle											
Instruction	1	2	3	4	5	6	7	8	9	10	11	12
BNEZ R0, loop	IF1	IF2	ID*	EX	MEM1	MEM2	WB		*forward R0 (ID* to *IF1) in cc3			
BRANCH+1		IF1	IF2	ID	EX	MEM1	MEM2	WB				
BRANCH+2			IF1	IF2	ID	EX	MEM1	MEM2	WB			
BRANCH TARGET				*IF1	IF2	ID	EX	MEM1	MEM2	WB		

Figure 2: Timing Diagram for A2 Case 1

	Clock Cycle												
Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13
previous instruction	IF1	IF2	ID	EX*	MEM1	MEM2	WB	*forward R0 (EX* to *ID) in cc4					
BNEZ R0, loop		IF1	IF2	stall	*ID**	EX	MEM1	MEM2	WB	**forward R0 (ID** to **IF) in cc5			
BRANCH+1			IF1	stall	IF2	ID	EX	MEM1	MEM2	WB			
BRANCH+2				stall	IF1	IF2	ID	EX	MEM1	MEM2	WB		
BRANCH TARGET						**IF1	IF2	ID	EX	MEM1	MEM2	WB	

Figure 3: Timing Diagram for A2 Case 2

Exercise Part (B) - Generate RISC Code for The Chacha20 Stream Cipher

(B1) The unoptimized RISC code for the first 3 lines of the QUARTER-ROUND operation is shown in Listing 1. The following assumptions are made for this RISC code:

- ASSUMPTION 1: assume that register R0 contains the address of the first word of the block of the initial key-stream
- ASSUMPTION 2: each word is 32 bits, so use LD and SD instead of LW and SD to load 32-bit words
- ASSUMPTION 3: words in the initial key-stream are stored consecutively in memory, address of second word is address of first world + 4 (bytes)
- ASSUMPTION 4: assume dual-ported memory that allows simultaneous read+read, read+write, write+write

Listing 1: Unoptimized RISC code for the first 3 lines of QUARTER-ROUND operation

```
1
2  % macros
3  #define Ra => R1; Rb => R2; Rd => R3     % define Ra, Rb, Rd
4
5  LD       Ra,(0)R0          % load a from memory
6  LD       Rb,(16)R0         % load b from memory
7  ADD      Ra,Ra,Rb          % a = a + b
8  SD       (0)R0,Ra          % store a in memory
9
10 LD       Ra,(0)R0          % load a from memory
11 LD       Rd,(48)R0         % load d from memory
12 XOR      Rd,Rd,Rd          % XOR(d,a)
13 SD       (48)R0,Rd         % store d in memory
14
15 LD       Rd,(48)R0         % load d from memory
16 ROT.L    Rd,Rd,#16         % ROTATE_LEFT(d, 16)
17 SD       (48)R0,Rd         % store d in memory
```

The timing diagram can be seen in Figure 4.

	Clock Cycle																									
Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
LD Ra,(0)R0	IF1	IF2	ID	EX	MEM1	MEM2	WB																			
LD Rb,(16)R0		IF1	IF2	ID	EX	MEM1	MEM2*	WB	*forward Rb (MEM2* to *EX) in cc7																	
ADD Ra,Ra,Rb			IF1	IF2	ID	stall	stall	*EX	MEM1	MEM2	WB															
SD (0)R0,Ra				IF1	IF2	stall	stall	ID	EX	MEM1	MEM2*	WB	*forward (0)R0 (MEM2* to *MEM1) in cc11													
LD Ra,(0)R0					IF1	stall	stall	IF2	ID	EX	stall	*MEM1	MEM2	WB												
LD Rd,(48)R0						stall	stall	IF1	IF2	ID	stall	EX	MEM1	MEM2*	WB	*forward Rd (MEM2* to *EX) in cc14										
XOR Rd,Rd,Rd									IF1	IF2	stall	ID	stall	stall	*EX	MEM1	MEM2	WB								
SD (48)R0,Rd										IF1	stall	IF2	stall	stall	ID	EX	MEM1	MEM2*	WB	*forward (48)R0 (MEM2* to *MEM1) in cc18						
LD Rd,(48)R0											stall	IF1	stall	stall	IF2	ID	EX	stall	*MEM1	MEM2*	WB	*forward Rd (MEM2* to *EX) in cc20				
ROT.L Rd,Rd,#16													stall	stall	IF1	IF2	ID	stall	stall	stall	*EX	MEM1	MEM2	WB		
SD (48)R0,Rd																IF1	IF2	stall	stall	stall	ID	EX	MEM1	MEM2	WB	

Figure 4: Timing Diagram for B1

(B2) The unoptimized RISC code for one QUARTER-ROUND operation (along with the associated stall cycles per instruction) is shown in Table 1. We make the same assumptions as we did in B1.

Table 1: Unoptimized RISC code for QUARTER-ROUND operation

Instruction	Number of stall cycles	Comment
LD Ra,(0)R0		load a from memory
LD Rb,(16)R0		load b from memory
ADD Ra,Ra,Rb		a = a + b
SD (0)R0,Ra		store a in memory
LD Ra,(0)R0		load a from memory
LD Rd,(48)R0		load d from memory
XOR Rd,Rd,Rd		XOR(d,a)
SD (48)R0,Rd		store d in memory
LD Rd,(48)R0		load d from memory
ROT.L Rd,Rd,#		ROTATE_LEFT(d, 16)
SD (48)R0,Rd		store d in memory
LW Rc,(32)R0		load c from memory
LW Rd,(48)R0		load d from memory
ADD Rc,Rc,Rd		c = c + d
SD (32)R0,Rc		store c in memory
LW Rb,(16)R0		load b from memory
LW Rc,(32)R0		load c from memory
XOR Rb,Rb,Rc		XOR(b,c)
SD (16)R0,Rb		store b in memory
LW Rb,(16)R0		load b from memory
ROT.L Rb,Rb,#		ROTATE_LEFT(b, 12)
SD (16)R0,Rb		store b in memory
LD Ra,(0)R0		load a from memory
LD Rb,(16)R0		load b from memory
ADD Ra,Ra,Rb		a = a + b
SD (0)R0,Ra		store a in memory
LD Ra,(0)R0		load a from memory
LD Rd,(48)R0		load d from memory
XOR Rd,Rd,Rd		XOR(d,a)
SD (48)R0,Rd		store d in memory
LD Rd,(48)R0		load d from memory
ROT.L Rd,Rd,#8		ROTATE_LEFT(d, 8)
SD (48)R0,Rd		store d in memory
LW Rc,(32)R0		load c from memory
LW Rd,(48)R0		load d from memory
ADD Rc,Rc,Rd		c = c + d

Continued on next page

Table 1 – continued from previous page

Instruction	Number of stall cycles	Comment
SD (32)R0,Rc		store c in memory
LW Rb,(16)R0		load b from memory
LW Rc,(32)R0		load c from memory
XOR Rb,Rb,Rc		XOR(b,c)
SD (16)R0,Rb		store b in memory
LW Rb,(16)R0		load b from memory
ROT.L Rb,Rb,#7		ROTATE_LEFT(b, 7)
SD (16)R0,Rb		store b in memory

(B3) hi

(B4) hi again

(B5)

(B6)

(B7)