Department of Electrical and Computer Engineering

The University of Texas at Austin

EE 460N, Fall 2016
Problem Set 4 Solutions
Yale N. Patt, Instructor
Siavash Zangeneh, Ali Fakhrzadehgan, Steven Flolid, Matthew Normyle, TAs

Problem Set 4 Solutions

1. 1. -3.25
   2. 1.25 x 2^(-128)
   3. Negative Infinity

2.
```
    | 4  3 | 6  2
-----------------
  0 | 0  0 | 0  0
  1 | 1  1 | 1  1
  2 | 2  2 | 2  0
  3 | 3  0 | 3  1
  4 | 0  1 | 4  0
  5 | 1  2 | 5  1
  6 | 2  0 | 0  0
  7 | 3  1 | 1  1
  8 | 0  2 | 2  0
  9 | 1  0 | 3  1
 10 | 2  1 | 4  0
 11 | 3  2 | 5  1
```

3. Multiplicand is 0011011110.

Cycle	Multiplier	Temp register
0	0001110010	0000000000----------
1	--00011100	000110111100--------
2	----000111	00000110111100------
3	------0010	1111001111011100----
4	--------00	000110001011011100--
5	----------	00000110001011011100

The result is 00000110001011011100.

4. The exponenet field has a value 63. Therefore the exponent is 63-63=0. The value is 1.0000001111111111 * 2^0. This value is 1.0156.

5.
```
The bias for the 1-8-23 IEEE format is 127.

  3EE00000h = 0 01111101 11000000000000000000000 = 1.11 * 2^-2
  3D800000h = 0 01111011 00000000000000000000000 = 1.0 * 2^-4

Adjusting exponents, we get

        1.11 * 2^(-2)
        0.01 * 2^(-2)


Since the exponents are now the same, we can add the two
fractions. The result has the same exponent as the operands. The
result is: 10.00 * 2^(-2). When normalized to IEEE format, this
becomes 1.00 * 2^(-1),

IEEE format: 0 01111110 00000000000000000000000 which in hex is
3F000000h.
```

6.
```
Model 0.001: 1-7-8
Smallest positive normalized number is 0 0000001 00000000 = 1.0 * 2^(1-63) = 2^-62
Largest postive normalized number is 0 1111110 11111111 = 1.11111111 * 2^(126-63) = 1.11111111 * 2^63
```

```
This model has (8+1) binary bits of precision. Since 1024 (2^10) is
approximately 1000 (10^3); 10 binary bits of precision correspond to
about 3 decimal digits of precision. Hence, this model has 9 * (3/10)
= 2.7 decimal digits of precision

Model 0.002: 1-5-10
Smallest positive normalized number is 0 00001 0000000000 = 1.0 * 2^(1-15) = 2^-14
Largest postive normalized number is 0 11110 1111111111 = 1.1111111111 * 2^(30-15) = 1.1111111111 * 2^15

This model has (10+1) binary bits of precision. Hence, this model has
11 * (3/10) = 3.3 decimal digits of precision

The model chosen would depend on the application - if a higher range
is desired, then model 0.001 is better; if precision is more
important, then model 0.002 will be preferred.
```

7. a. 5 bits
   b. 1

   c.

```
      48        0 110 10000
      19.5      0 101 00111
      5/16      0 000 01010
      0         0 000 00000
     -1         1 001 00000
   -infinity    1 111 00000
```

8. The truth table to implement the logic is as follows

4LSB of Multiplier	1st stage LSHF	operation performed	2nd stage LSHF	Borrow-Out	Total Bits processed
0000	0	N/A	4	0	4
0001	0	ADD	4	0	4
0010	1	ADD	3	0	4
0011	0	SUB	2	1	2
0100	2	ADD	2	0	4
0101	0	ADD	2	0	2
0110	1	SUB	2	1	3
0111	0	SUB	3	1	3
1000	3	ADD	1	0	4
1001	0	ADD	3	0	3
1010	1	ADD	2	0	3
1011	0	SUB	2	1	2
1100	2	SUB	2	1	4
1101	0	ADD	2	0	2
1110	1	SUB	3	1	4
1111	0	SUB	4	1	4

Average number of bits processed in a cycle = (4+4+4+2+4+2+3+3+4+3+3+2+4+2+4+4)/16 = 3.25

9. Assume an ADD operation is executed like this in the pipeline:

1	2	3	4	5	6	7
F	D	A	A	A	A	S

and a MUL operation is executed like this in the pipeline:

1	2	3	4	5	6	7	8	9
F	D	M	M	M	M	M	M	S

F: Fetch, D: Decode, A: Execute stage (for ADD), M: Execute stage for MUL, S: Store result (Write-back)

a. ADDs require 7 cycles (fetch, decode, 4 execute, store), and MULs require 9 cycles (fetch, decode, 6 execute, store). For 3 ADD instructions and 3 MUL instructions, the execution time is **3 × 7 + 3 × 9 = 48** cycles.
b. Pipeline with scoreboarding and five adders and five multipliers (assuming one instruction fetched per cycle):

   i. No data forwarding: the destination register is marked valid in the S stage (a dependent instruction starts executing after the S stage of the instruction it depends on)

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
F	D	M	M	M	M	M	M	S																	
	F	D	D	D	D	D	D	D	A	A	A	A	S												
		F	F	F	F	F	F	F	D	A	A	A	A	S											
									F	D	M	M	M	M	M	M	S								

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
										F	D	D	D	D	D	D	D	A	A	A	A	S			
											F	F	F	F	F	F	F	D	M	M	M	M	M	M	S

Execution time: 26 cycles

ii. With forwarding:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
F	D	M	M	M	M	M	M	S															
	F	D	D	D	D	D	D	A	A	A	A	S											
		F	F	F	F	F	F	D	A	A	A	A	S										
								F	D	M	M	M	M	M	M	S							
									F	D	D	D	D	D	D	A	A	A	A	S			
										F	F	F	F	F	F	D	M	M	M	M	M	M	S

Execution time: 24 cycles

c. Pipeline with scoreboarding and one adder and one multiplier (assuming one instruction fetched per cycle):

i. The adder and multiplier are not pipelined and there is no data forwarding:

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	27	28	29
F	D	M	M	M	M	M	M	S																				
	F	D	D	D	D	D	D	D	A	A	A	A	S															
		F	F	F	F	F	F	F	D	D	D	D	A	A	A	A	S											
									F	F	F	F	D	M	M	M	M	M	M	S								
													F	D	D	D	D	D	D	D	A	A	A	A	S			
														F	F	F	F	F	F	F	D	M	M	M	M	M	M	S

Execution time: 29 cycles

ii. The adder and multiplier are not pipelined and there is data forwarding:

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	27
F	D	M	M	M	M	M	M	S																		
	F	D	D	D	D	D	D	A	A	A	A	S														
		F	F	F	F	F	F	D	D	D	D	A	A	A	A	S										
									F	F	F	D	M	M	M	M	M	M	S							
												F	D	D	D	D	D	D	A	A	A	A	S			
													F	F	F	F	F	F	D	M	M	M	M	M	M	S

Execution time: 27 cycles

iii. The adder and multiplier are pipelined and there is no data forwarding:

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
F	D	M	M	M	M	M	M	S																	
	F	D	D	D	D	D	D	D	A	A	A	A	S												
		F	F	F	F	F	F	F	D	A	A	A	A	S											
									F	D	M	M	M	M	M	M	S								
										F	D	D	D	D	D	D	D	A	A	A	A	S			
											F	F	F	F	F	F	F	D	M	M	M	M	M	M	S

Execution time: 26 cycles

iv. The adder and multiplier are pipelined and there is data forwarding:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
F	D	M	M	M	M	M	M	S															
	F	D	D	D	D	D	D	A	A	A	A	S											
		F	F	F	F	F	F	D	A	A	A	A	S										
								F	D	M	M	M	M	M	M	S							
									F	D	D	D	D	D	D	A	A	A	A	S			
										F	F	F	F	F	F	D	M	M	M	M	M	M	S

Execution time: 24 cycles

10. Assumptions:

- There are enough ports to the register file.

- There are enough ports to the memory.
- There are separate execution units for ADD, AND, STW, and BR instructions (They can all be in the execute stage at the same time.)

a. Pipeline diagram:

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Start of first iteration (R1 is even)															
`STW`	F	D	E	E	E	S									
`ADD`		F	D	E	E	E	S								
`AND`			F	D	E	E	S								
`BRz`				F	D	D	E	S							
`ADD`								F	D	E	E	E	S		
`ADD`									F	D	E	E	E	S	
`BRp`										F	D	D	D	E	S
End of the first iteration (R1 is odd now)															
`STW`															F

The loop takes the same number of cycles to execute for even and odd values of R1. Each iteration takes 14 cycles in the steady state. There are 5 iterations for even values of R1 and 4 iterations for odd values of R1. The total number of cycles is:

$(14 \times 5) + (14 \times 4) + 1 = 127$

The extra 1 cycle comes from the last iteration (Store result stage of the BRp instruction).

b. Pipeline diagram: Note that predicted branches are decoded immediately, and then stalled in EXECUTE stage until branch dependcy is resolved.

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Start of first iteration (R1 is even)																
`STW`	F	D	E1	E2	E3	S										
`ADD`		F	D	E1	E2	E3	S									
`AND`			F	D	E1	E2	S									
`BRz`				F	D	E1	E1	S								
`ADD`					F	D	E1	E2	E3	S						
`ADD`						F	D	E1	E2	E3	S					
`BRp`							F	D	E1	E1	E1	S				
End of the first iteration (R1 is odd now)																
`STW`								F	D	E1	E2	E3	S			

The loop takes the same number of cycles to execute for even and odd values of R1. Each iteration takes 12 cycles but 5 cycles can be overlapped with the next iteration. The total number of cycles is:

$(7 \times 9) + 5 = 68$

c. BRz instruction will always be predicted not taken. It is taken when R1 is even. So it will be mispredicted when R1 is even and correctly predicted when R1 is odd. The following diagram shows three consecutive iterations of the loop. In the first iteration, BRz is mispredicted, in the second iteration it is correctly predicted.

The first BRp instruction is always predicted taken. It is always predicted correctly. The second BRp instruction is also always predicted taken. It is mispredicted only once in the last iteration of the loop.

```
        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  27  28  29  30  3
------------------------------------------------------------------------------------------------------------- Start of first it
STW     F | D | E | E | E | S
ADD         F | D | E | E | E | S
AND             F | D | E | E | S
BRz                 F | D |   | E | S  (Mispredicted)
ADD R1, R1, #3          F | D | E | Flushed
ADD R5, R5, #-1             F | D | Flushed
BRp DOIT                        F | Flushed
ADD R1, R1, #1                      F | D | E | E | E | S
ADD R7, R7, #-1                         F | D | E | E | E | S
BRp DOIT                                    F | D |   |   | E | S (Correctly predicted)
------------------------------------------------------------------------------------------------------------ Start of second it
STW                                             F | D | E | E | E | S
ADD                                                 F | D | E | E | E | S
AND                                                     F | D | E | E | S
BRz                                                         F | D |   | E | S (Correctly predicted)
ADD R1, R1, #3                                                  F | D | E | E | E | S
ADD R5, R5, #-1                                                     F | D | E | E | E | S
BRp DOIT                                                                F | D |   |   | E | S (Correctly predicted)
----------------------------------------------------------------------------------------------------------- Third iteration (R1
STW                                                                         F | D | E | E | E | S
ADD                                                                             F | D | E | E | E | S
AND                                                                                 F | D | E | E | S
BRz                                                                                     F | D |   | E | S  (Mispredicted)
ADD R1, R1, #3                                                                              F | D | E | Flushed
ADD R5, R5, #-1                                                                                 F | D | Flushed
BRp DOIT                                                                                            F | Flushed
ADD R1, R1, #1                                                                                          F | D | E | E | E | S
ADD R7, R7, #-1                                                                                      F | D | E | E | E |
BRp DOIT                                                                                                 F | D |   |   |
       <--------- beginning of the loop ------><---------------- steady state (17 cycles) --------------------->
```

Loop steady state is shown above. It takes 17 cycles and it is repeated 4 times. The beginning of the loop (until the steady state) takes 10 cycles as shown above. The end of the loop (part of the last iteration which is not in steady state) takes 5 more cycles to execute. The total number of cycles is:

10 + (4 × 17) + 5 = 83

Prediction accuracies for each branch are:

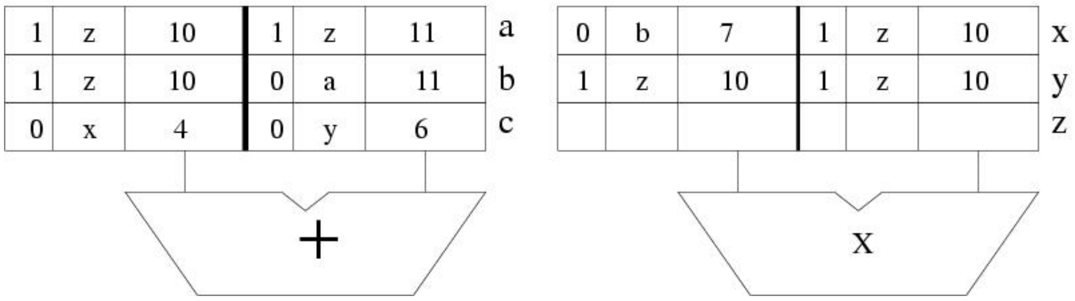
- BRz = 4/9 (Correctly predicted when R1 is odd, R1 is odd for 4 iterations of the loop)
- first BRp = 4/4
- second BRp = 4/5 (Mispredicted only in the last iteration - Note that this misprediction does not affect the number of cycles it takes to execute the loop)

Combined branch prediction accuracy = 12/18 = 67%

11. 1.

1	ADD R7, R6, R7
2	ADD R3, R6, R7
3	MUL R0, R3, R6
4	MUL R2, R6, R6
5	ADD R2, R0, R2

2.



3.

	V	tag	value
R0	0	x	4
R1	1	z	5
R2	0	c	6
R3	0	b	7
R4	1	z	8
R5	1	z	9
R6	1	z	10
R7	0	a	11

12. Assume the destination register is indicated first, followed by the source(s).

(As stated in the question, cycle counts assume a 16-way interleaved memory so that a new access can be started each cycle. Also, the adder and multiplier are pipelined.)

1. Scalar processor (one of the possible solutions)

```
         MOVI   R0, 0           (1 cycle)
         LEA    R4, A           (1 cycle)
         LEA    R5, B           (1 cycle)
         LEA    R6, C           (1 cycle)
         LEA    R7, D           (1 cycle)
   LOOP  LD     R1, R5, R0      (11 cycles)
         LD     R2, R6, R0      (11 cycles)
         LD     R3, R7, R0      (11 cycles)
         MUL    R1, R1, R2      (6 cycles)
         ADD    R1, R1, R3      (4 cycles)
         RSHFA  R1, R1, 1       (1 cycle)
         ST     R1, R4, R0      (11 cycles)
         ADD    R0, R0, 1       (4 cycles)
         ADD    R1, R0, -100    (4 cycles)
         BNZ    LOOP            (1 cycle)
```

5 × 1 + 100 × (11 + 11 + 11 + 6 + 4 + 1 + 11 + 4 + 4 + 1) = 6405 cycles

2. Vector Processor:

The loop could be split into two parts as 64 and 36. Assume the vector code looks as follows: (This solution assumes that the addressing mode VLD V0, B+64 exists - if it doesn't, then you would need 4 + 3 cycles using a pipelined adder to add 64 to A,B,C,D)

```
LD     Vln, #64           (1 cycle)
LD     Vst, #1            (1 cycle)
VLD    V0, B              (11 + 63 cycles)
VLD    V1, C              (11 + 63 cycles)
Vmul   V2, V0, V1         (6 + 63 cycles)
VLD    V3, D              (11 + 63 cycles)
Vadd   V4, V2, V3         (4 + 63 cycles)
Vrshfa V5, V4, 1          (1 + 63 cycles)
VST    V5, A              (11 + 63 cycles)
LD     Vln, #36           (1 cycle)
VLD    V0, B+64           (11 + 35 cycles)
VLD    V1, C+64           (11 + 35 cycles)
Vmul   V2, V0, V1         (6 + 35 cycles)
VLD    V3, D+64           (11 + 35 cycles)
Vadd   V4, V2, V3         (4 + 35 cycles)
Vrshfa V5, V4, 1          (1 + 35 cycles)
VST    V5, A+64           (11 + 35 cycles)
```

1. Vector processor without chaining (vector instructions done serially)

   First part:

   ```
   |-1-|-1-|--11--|---63---|--11--|---63---|-6-|---63---|--11--|---63---|-4-|---63---|-1-|---63---|--11--|
   ```

   Second part:

   ```
   |-1-|--11--|--35--|--11--|--35--|-6-|--35--|--11--|--35--|-4-|--35--|-1-|--35--|--11--|--35--|
   ```

   2 + (74 × 4 + 69 + 67 + 64) + 1 + (46 × 4 + 41 + 39 + 36) = 799 cycles

2. Vector processor with chaining, 1 port to memory.

   Chaining means the machine begins the next operation as soon as the operands are ready.

   ```
   |-1-|-1-|--11--|---63---|--11--|---63---|--11--|---63---|--11--|---63---|--11--|--35--|--11--|--35--|-
                                  |-6-|---63---|  |-4-|---63---|                                      |-6-|--35
                                                      |-1-|---63---|
                                                             |-1-|
   ```

   Chaining, in this instance, hides the VMULT, VADD, and VSHL operations. Memory becomes the primary bottleneck.

   482 cycles

3. Vector processor with chaining; 2 loads, 1 store per cycle.

   ```
   |-1-|-1-|--11--|---63---|       VLD V0, B
            |--11--|---63---|       VLD V1, C
                   |-6-|---63---|
                           |--11--|---------------63-----------------|     VLD V3, D  Needs to wait until
                                  |-4-|---63---|
                                      |-1-|---63---|
                                          |--11--|----------------63-----------------|      VST V5,A
                                            |-1-|--11--|---35---|         VLD V0, B+64  (this finishes soo
                                                                |--11--|----35----|           VLD V1, C+6
                                                                       |-6-|----35----|
                                                                           |--11--|--35--|
                                                                                  |-4-|--35--|
                                                                                      |-1-|--35--|
                                                                                          |--11--|--35--|
   ```

   #cycles = 1 + 1 + (11 + 63) + 11 + 4 + 1 + 1 + 1 + (11 + 35) + 11 + 6 + 11 + 4 + 1 + (11 + 35) = 219 cycles

Another solution is to split the loop into two equal parts as 50 and 50.

```
LD     Vln, #50           (1 cycle)
LD     Vst, #1            (1 cycle)
VLD    V0, B              (11 + 49 cycles)
VLD    V1, C              (11 + 49 cycles)
Vmul   V2, V0, V1         (6 + 49 cycles)
VLD    V3, D              (11 + 49 cycles)
Vadd   V4, V2, V3         (4 + 49 cycles)
Vrshfa V5, V4, 1          (1 + 49 cycles)
VST    V5, A              (11 + 49 cycles)
VLD    V0, B+50           (11 + 49 cycles)
VLD    V1, C+50           (11 + 49 cycles)
Vmul   V2, V0, V1         (6 + 49 cycles)
VLD    V3, D+50           (11 + 49 cycles)
Vadd   V4, V2, V3         (4 + 49 cycles)
Vrshfa V5, V4, 1          (1 + 49 cycles)
VST    V5, A+50           (11 + 49 cycles)
```

1. Vector processor without chaining (vector instructions done serially)

First part:

```
|-1-|-1-|--11--|---63---|--11--|---63---|-6-|---63---|--11--|---63---|-4-|---63---|-1-|---63---|--11--
```

Second part:

```
|--11--|--35--|--11--|--35--|-6-|--35--|--11--|--35--|-4-|--35--|-1-|--35--|--11--|--35--|
```

2 + (74 × 4 + 69 + 67 + 64) + (46 × 4 + 41 + 39 + 36) = 798 cycles

2. Vector processor with chaining, 1 port to memory. This part for solution B will be the same as solution B. Total = 482 cycles

3. Vector processor with chaining; 2 loads, 1 store per cycle

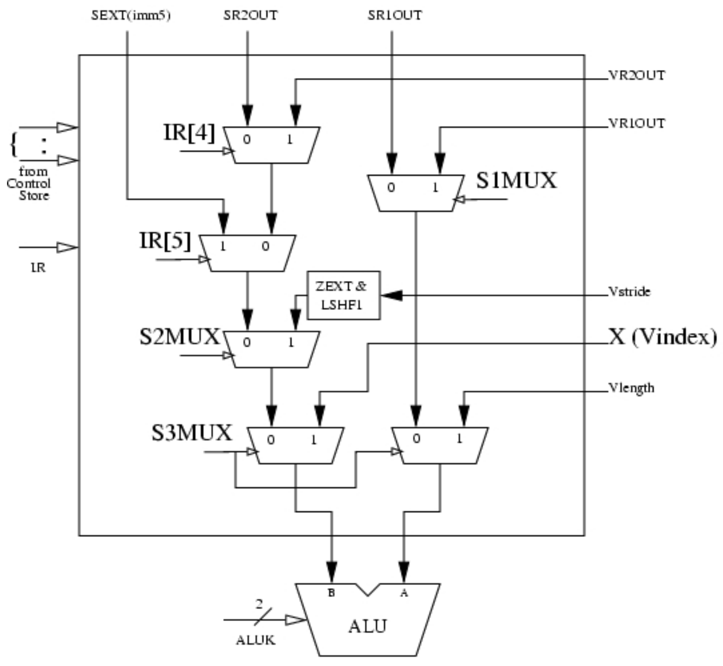
```
|-1-|-1-|-11-|-----49-----|
         |-11-|-----49-----|
              |-6-|-----49-----|
                          |-11-|-----49-----|   ** LD D would need to wait for LD B to finish
                               |-4-|-----49-----|
                                   |-1-|-----49-----|
                                       |-11-|-----49-----|
                                        |-11-|-----49-----|
                                            |-11-|-----49-----|  ** LD C+50 would need to wait till t
                                                 |-6-|-----49-----|
                                                          |-11-|-----49-----|  ** LD D+50 would need t
                                                               |-4-|-----49-----|
                                                                   |-1-|-----49-----|

                                                                       |-11-|-----49-----|
```

1 + 1 + (11 + 49) + 11 + 4 + 1 + 1 + (11 + 49) + 11 + 4 + 1 + (11 + 49) = 215 cycles

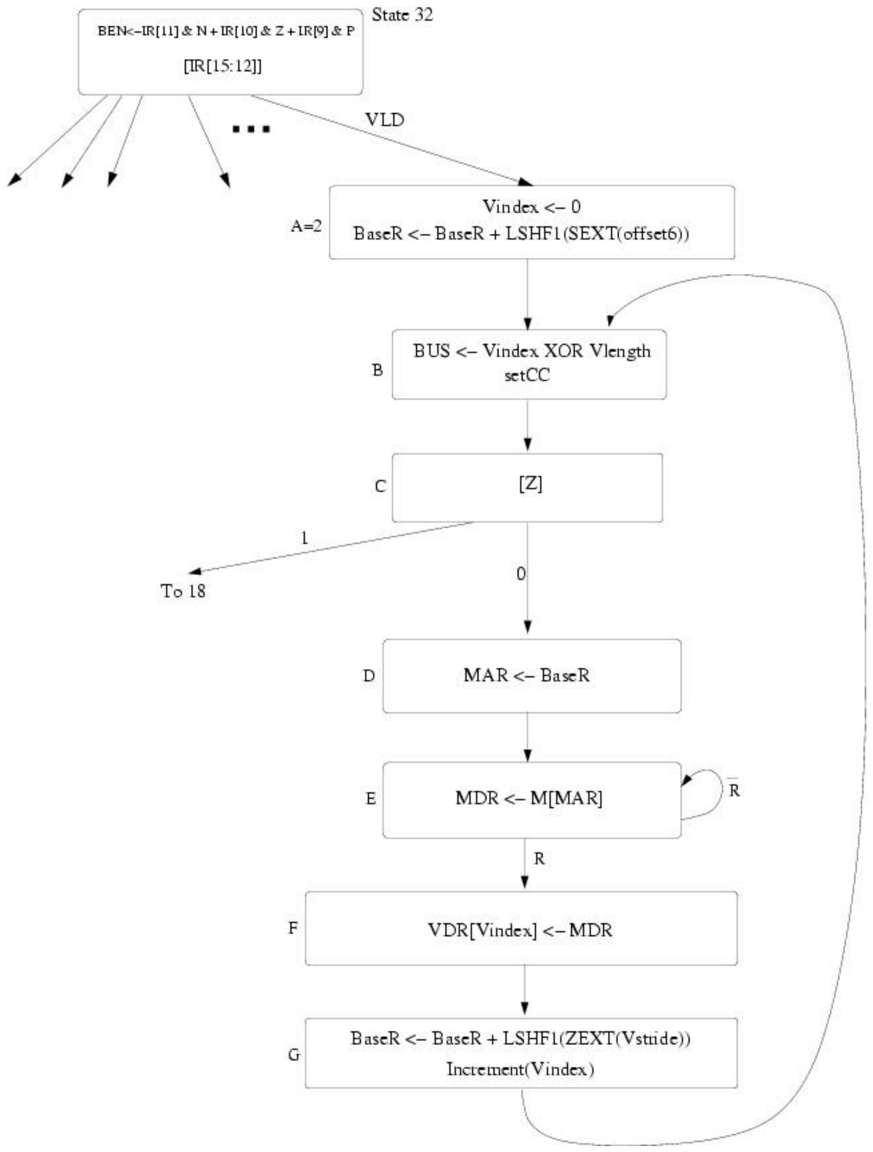
13. 1. X is the Vector Index Register, which holds the pointer to the element that is being processed by the vector instruction.

2. The two control signals are RESET.VINDEX and INCREMENT.VINDEX

3.



4. The following solution assumes a change to the microsequencer to branch according to the Z condition code.


State 32
BEN<-IR[11] & N + IR[10] & Z + IR[9] & P
[IR[15:12]]
...
VLD
A=2
Vindex <- 0
BaseR <- BaseR + LSHF1(SEXT(offset6))
B
BUS <- Vindex XOR Vlength
setCC
C
[Z]
1
To 18
0
D
MAR <- BaseR
E
MDR <- M[MAR]
$\overline{R}$
R
F
VDR[Vindex] <- MDR
G
BaseR <- BaseR + LSHF1(ZEXT(Vstride))
Increment(Vindex)

14. This problem has been postponed to problem set 5
15. This problem has been postponed to problem set 5
16. This problem has been postponed to problem set 5