# National University of Computer and Emerging Sciences, Lahore Campus

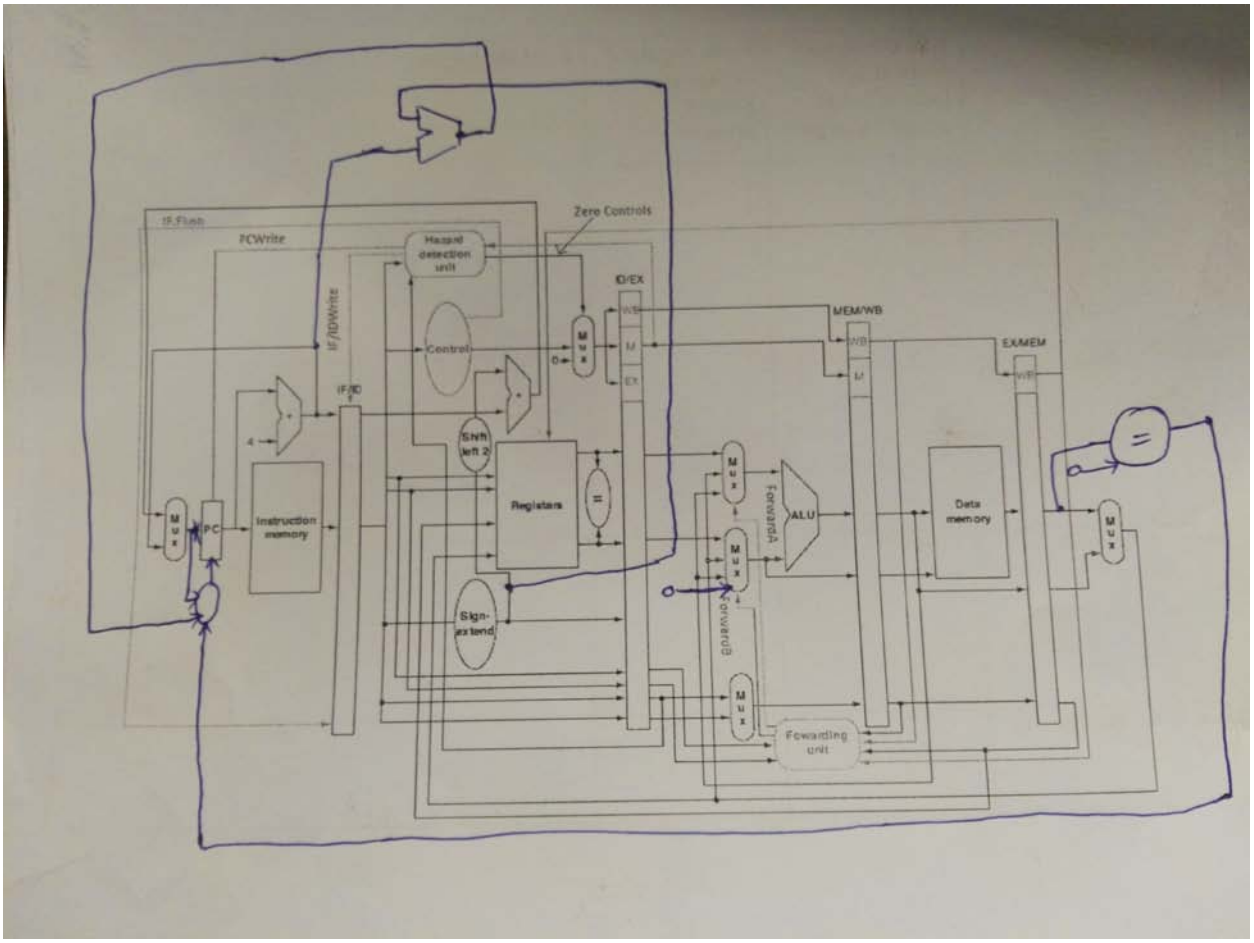
| | | | |
|---|---|---|---|
| Course: | Computer Architecture | Course Code: | EE204 |
| Program: | BS(Computer Science) | Semester: | Fall 2016 |
| Duration: | 150 Minutes | Total Marks: | 70 |
| Paper Date: | 19-12-2016 | Weight | 45 |
| Section: | All Sections | Page(s): | 8 |
| Exam: | Final Solution | | |

**Note: Only Attempt question on the provided space. Rough sheet may be used but it should not be attached.**

**Q1:** Suppose we want to add support for these two instructions in our MIPS architecture. Marks (5 +2+3)

| | |
|---|---|
| **bez (Rs), Label** | **If Mem[Rs] == 0 then PC = PC + Offset** |

a. What must be changed in the pipelined datapath to add this instruction to the MIPS Instruction Set Architecture? Make the changes to the following diagram to answer this question. Use space provided above the diagram to draw any new required resources/stages. You can draw lines from/to the diagram for input/output of these units. Draw neatly so that its easier for the instructor to understand your logic.



b. Which new control signals ( if any) must be added to the pipeline? Name the signal and briefly describe its purpose in the table below. Please note that the number of rows (number of control signals) does not depict the required control signals for required changes. Required signals may be less or greater the rows.

| **Control Signal** | **Description** |
|---|---|
| Signal to determine bezi instruction | This signal must be ANDed with the zero comparison of WB stage. |
| | |
| | |

c. Which of the following hazards(if any) will occur if the above mentioned new instruction is added to the data path? If any hazard occurs, then how many stalls will be required to avoid any wrong execution of the code segments? Answer the question by filling the following table.

| **Hazard Type** | **Will the hazard occur if new instruction is added? Yes/No** | **Stalls required in case of hazard occurring** | **Brief Description** |
|---|---|---|---|
| Data Hazard | NO | | |
| Control Hazard | Yes | **5** | Decision is made at the $5^{th}$ stage so next instruction cannot be fetched before the completion of this instruction. |
| Structural Hazard | No | | |
| None of the hazards will occur | | | |

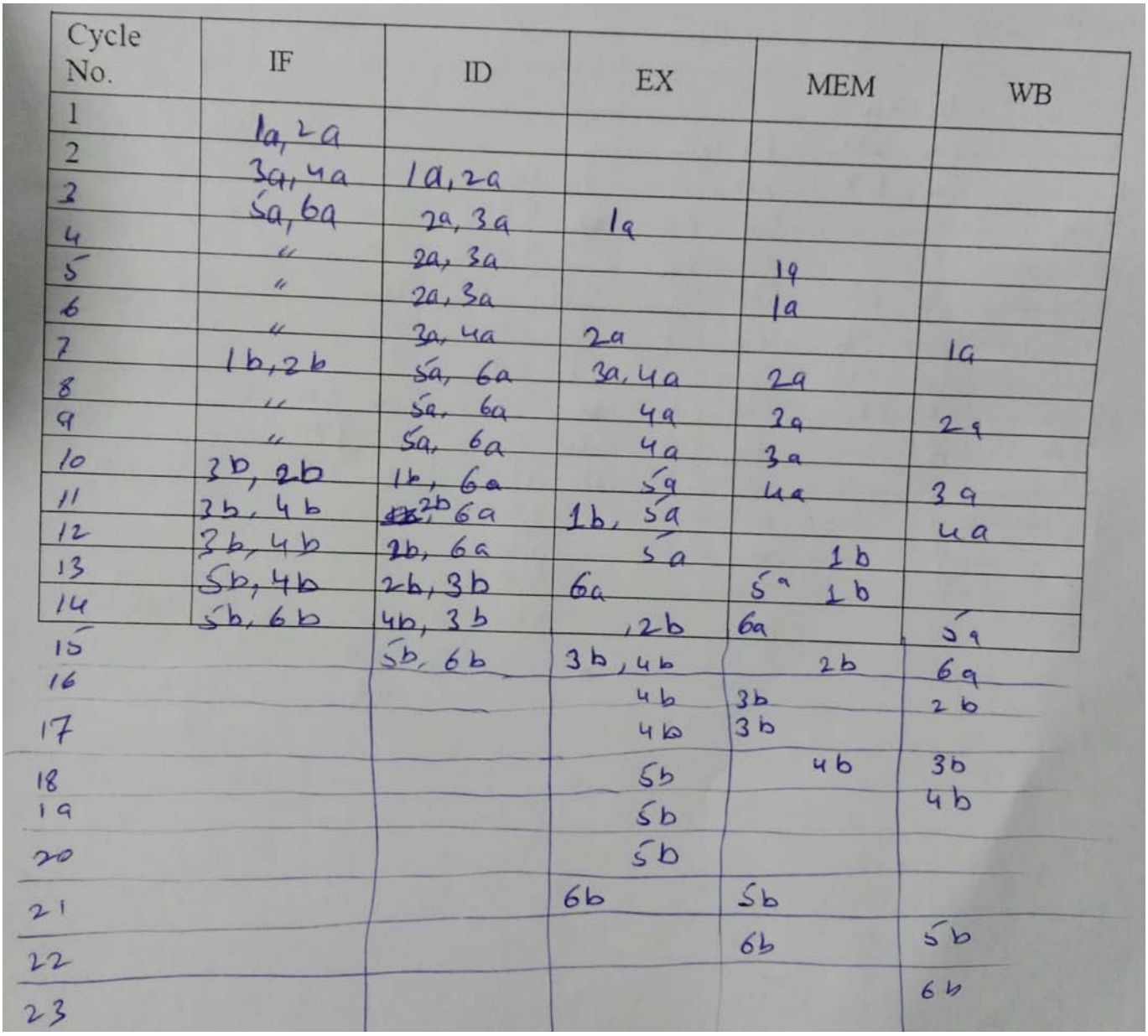
**Q2:** Suppose we want to execute the following program on a 2-way superscalar processor(**Dynamic multiple Issue processor**) with three execution units that have the following latencies. (5+5+5) marks

**Adder 1 cycles**
**Multiplier 3 cycles**
**Divider 5 cycles**
**Load/Store operation in the memory stage takes 2 cycles.**

```
Loop:  LD R1, 40(R6)        // Instruction#1
       ADD  R5, R5, R1      // Instruction#2
       ST R1, 20(R5)        // Instruction#3
       MUL R6, R6, 2        // Instruction#4
       MUL R5, R5, -2       // Instruction#5
       BEQ R5, R0, Loop     // Instruction#6
```

You have to show the execution of two iterations of this loop on a 2-way superscalar processor. You can move an instruction from one stage of the processor to the other as soon as a slot is available in the next stage. Instruction wait in the decode stage in case there is any dependency. Buffers are available after execution and memory stages to hold completed operations. Forwarding is implemented.

a. In-order issue and out-of-order execution rules are applied. For each cycle, show which instructions in at which stage by filling the following table. You don't need to write complete instructions. Just write the instruction numbers from the code above. Remember you cannot re-arrange the instructions for this part of the question.



| Cycle No. | IF | ID | EX | MEM | WB |
|---|---|---|---|---|---|
| 1 | 1a, 2a | | | | |
| 2 | 3a, 4a | 1a, 2a | | | |
| 3 | 5a, 6a | 2a, 3a | 1a | | |
| 4 | " | 2a, 3a | | 1a | |
| 5 | " | 2a, 3a | | 1a | |
| 6 | " | 3a, 4a | 2a | | 1a |
| 7 | 1b, 2b | 5a, 6a | 3a, 4a | 2a | |
| 8 | " | 5a, 6a | 4a | 2a | 2a |
| 9 | " | 5a, 6a | 4a | 3a | |
| 10 | 3b, 2b | 1b, 6a | 5a | 4a | 3a |
| 11 | 3b, 4b | 2b 6a | 1b, 5a | | 4a |
| 12 | 3b, 4b | 2b, 6a | 5a | 1b | |
| 13 | 5b, 4b | 2b, 3b | 6a | 5a 1b | |
| 14 | 5b, 6b | 4b, 3b | 2b | 6a | 5a |
| 15 | | 5b, 6b | 3b, 4b | 2b | 6a |
| 16 | | | 4b | 3b | 2b |
| 17 | | | 4b | 3b | |
| 18 | | | 5b | 4b | 3b |
| 19 | | | 5b | | 4b |
| 20 | | | 5b | | |
| 21 | | | 6b | 5b | |
| 22 | | | | 6b | 5b |
| 23 | | | | | 6b |

b. Now unroll the loop for level-2 (2 iterations only) and reschedule(rearrange) the instructions to get best execution time on a 2-way superscalar processor with the specifications described above.

1a
4a
2a
3a
5a
1b
4b
2b
3b
5b
6

c. Show the execution of this sequence of instruction on 2-way superscalar processor for In-order issue and out-of-order execution and completion. Don't write whole instructions instead use the Instruction number assigned in part a.

| Cycle No. | IF | ID | EX | MEM | WB |
|---|---|---|---|---|---|
| 1 | 1a, 4a | | | | |
| 2 | 2a, 3a | 1a, 4a | | | |
| 3 | 5a, 1b | 2a, 3a | 1a, 4a | | |
| 4 | " | 2a, 3a | 4a | 1a | |
| 5 | " | 2a, 3a | 4a | 1a | |
| 6 | 4b, 1b | 5a, 3a | 2a, | 4a | 1a |
| 7 | 2b, 3b | 4b, 1b | 3a, 5a | 2a | 4a |
| 8 | 5b, 3b | 4b, 2b | 1b, 5a | 3a | 2a |
| 9 | 5b, 3b | 4b, 2b | 5a | 3a 1b | |
| 10 | 6 | 3b, 5b | 2b, 4b | 5a, | 3a |
| 11 | | 6, 5b | 3b, 4b | 2b, | 5a, 1b |
| 12 | | 6, 5b | 4b | 3b | 2b |
| 13 | | 6 | 5b | 4b 3b | |
| 14 | | 6 | 5b | | 4b, 3b |
| 15 | | 6 | 5b | 5b | |
| 16 | | | 6 | 6 | 5b |
| 17 | | | | | 6 |

**Q3:**Suppose we have virtual memory in a system with page size of 4KB, Virtual address is 16 bits and physical address is 15 bits. TLB cache is fully associative with 4 entries and least recently used replacement policy. L1 cache has 4 entries. Following is the snapshot for Page table, TLB cache and L1 cache. Marks 15

**Page table:**

| | Valid | Physical Page # |
|---|---|---|
| 0000 | 0 | DISK |
| 0001 | 1 | 001 |
| 0010 | 1 | 000 |
| 0011 | 1 | 010 |
| 0100 | 0 | DISK |
| 0101 | 0 | DISK |
| 0110 | 0 | DISK |
| 0111 | 0 | DISK |
| 1000 | 0 | DISK |
| 1001 | 1 | 110 |
| 1010 | 1 | 011 |
| 1011 | 1 | 101 |
| 1100 | 1 | 111 |
| 1101 | 0 | DISK |
| 1110 | 1 | 100 |
| 1111 | 0 | DISK |

**TLB cache:**

| Valid | LRU | Tag | Physical Page # |
|---|---|---|---|
| 1 | 3 | 1110 | 100 |
| 1 | 4 | 0011 | 010 |
| 1 | 2 | 1001 | 110 |
| 1 | 1 | 1100 | 111 |

**L1 cache:**

| Valid | Tag | DATA |
|---|---|---|
| 1 | 11001010 | 128 bytes data |
| 1 | 00100110 | 128 bytes data |
| 1 | 01011010 | 128 bytes data |
| 1 | 11101001 | 128 bytes data |

a. For each of the following addresses write the corresponding physical addresses and tell whether it is TLB hit/miss, L1 hit/miss or page fault.

| | Virtual address | Physical address | TLB hit/miss | L1 hit/miss | Page fault |
|---|---|---|---|---|---|
| A | 1110 0010 0010 0100 | 100001000100100 | Hit | Miss | No |
| B | 0111 1010 0110 0000 | PF | Miss | Miss | Yes |
| C | 1110 0101 0001 0001 | 100010100010001 | Hit | Miss | No |
| D | 1100 0001 1100 0101 | 111000111000101 | Hit | Miss | No |
| E | 1000 1001 1010 0011 | PF | Miss | Miss | Yes |

b. In case of a TLB miss, new entry comes in the TLB cache replacing the least recently used value. Value of 1 in LRU in above table represents the most recently used value and 4 represent the least recently used value. If the above addresses are accessed in same sequence, what would be the final state of TLB cache?

| Valid | LRU | Tag | Physical Page # |
|---|---|---|---|
| 1 | 3 | 1110 | 100 |
| 1 | 4 | 0111 | 000 |
| 1 | 1 | 1000 | 001 |
| 1 | 2 | 1100 | 111 |

**Note:** In case of a page fault, you can assign a new physical address to that specific virtual address in above Page table.

**Q4:** Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2. Marks (3 + 3 + 4)

a. Which processor has the highest performance? Express in the form of instructions per second?

b. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions of the program.
Number of Cycles: _____

c. We are trying to reduce the execution time by 30% but this leads to an increase of20% in the CPI. What clock rate should we have to get this time reduction?

Solution:

A)

Processor: P2

P1 = $2 * 10^9$ instr/s

P2 = $2.5 * 10^9$ instr/s

P3 = $1.82 * 10^9$ instr/s

B)

Number of cycles:

$3 * 10^{10}$

$2.5 * 10^{10}$

$4 * 10^{10}$

Number of instructions:

$2 * 10^{10}$

$2.5 * 10^{10}$

$1.82 * 10^{10}$

C)

Clock Rate:

5.13

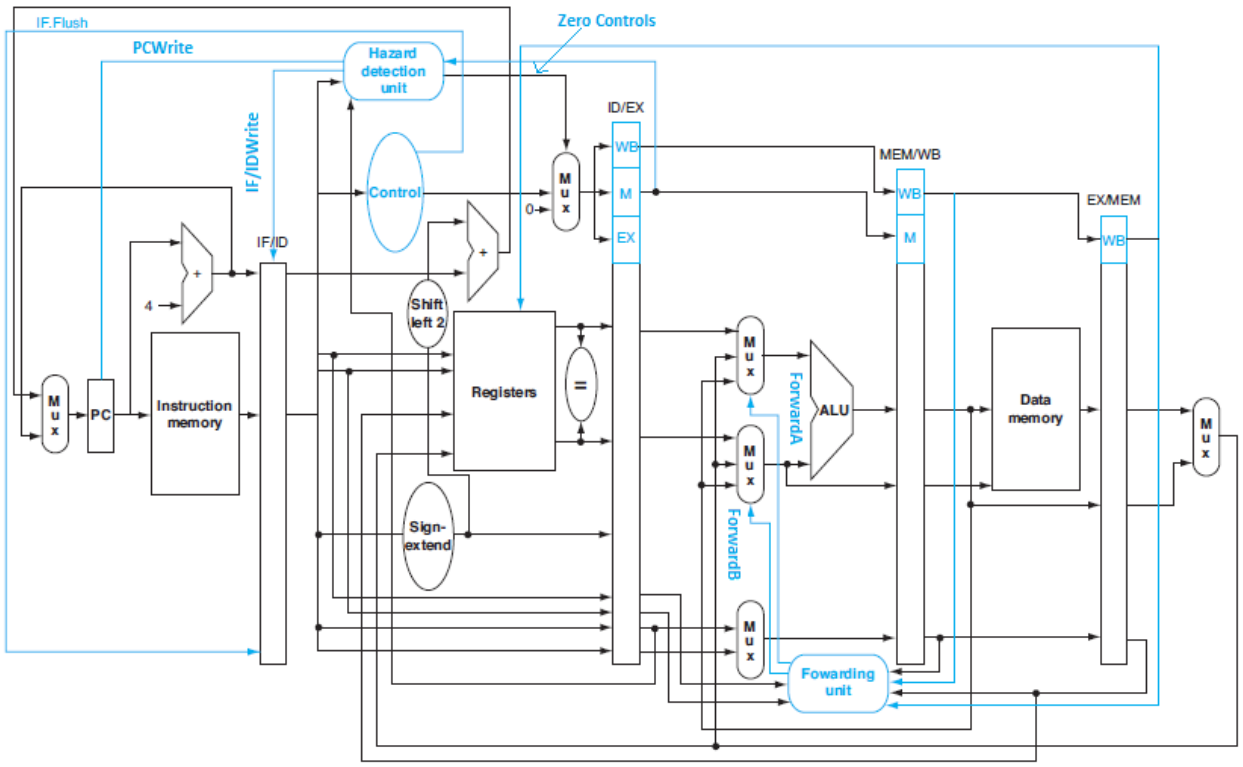
4.27

6.84

**Q5.** Consider the following code segment Marks: 20

```
        lw R1, 12 (R2)
        add R3,R1,R4
        lw R5, 16(R3)
        beq R5,12(R2) // Here the branch is taken i.e. Values of R5 and 12(R2) are equal.
        sw R5, 12(R2)
Label:  add R5,R1,R3
        sw R5, 12(R2)
```

Consider all the changes incorporated for reducing the stalls occurring due to Data hazards and Control hazards in the 5-stage pipelined processor. Refer to figure below for the finalized Data path of the processor. Also consider that 1-bit predictor is used for branch predictions. Predictor always predicts that the branch is **not taken**. What will be the values of the controls signals, forwarding and stall signals when the code segment provided above is executed on this processor. Fill the table provided on the next page to answer the question. Please note that the number of rows (number of clock cycles) does not depict the required clock cycles for the above code. Required Clock Cycles may be less or greater the mentioned ones. Draw more rows yourself if you need.




Solution:

| | Reg Dst | ALU Src | Mem to Reg | Reg Write | Mem Read | Mem Write | Branch | ALU 1Op | ALU 2Op | Forward A | Forward B | Pcwrite | IDWrite/IF | Flush.IF | Zero Controls |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clock Cycle 1 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Clock Cycle 2 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | – | – | 0 | 0 | 0 | 0 |
| Clock Cycle 3 | x | x | x | x | x | x | x | x | x | 0 0 | 0 0 | 1 | 1 | 0 | 1 |
| Clock Cycle 4 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | – | – | 1 | 1 | 0 | 1 |
| Clock Cycle 5 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 1 | 0 0 | 0 | 0 | 0 | 0 |
| Clock Cycle 6 | – | – | – | – | – | – | – | – | – | 0 1 | 0 0 | 0 | 0 | 0 | 0 |
| Clock Cycle 7 | – | – | – | – | – | – | – | – | – | – | – | – | – | 0 | – |
| Clock Cycle 8 | x | 0 | x | 0 | 0 | 0 | 1 | 0 | 1 | 0 0 | 0 0 | 1 | 1 | 0 | 1 |
| Clock Cycle 9 | – | – | – | – | – | – | – | – | – | – | – | 1 | 1 | 1 | 1 |
| Clock Cycle 10 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | – | – | 1 | 1 | 0 | 1 |
| Clock Cycle 11 | x | 1 | x | 0 | 0 | 1 | 0 | 0 | 0 | 0 0 | 0 0 | x | x | 0 | x |
| Clock Cycle 12 | – | – | – | – | – | – | – | – | – | 0 1 | 0 0 | x | x | 0 | x |