CO224 – PROCESSOR-BUILDING A MEMORY HIERARCHY

Group: 09

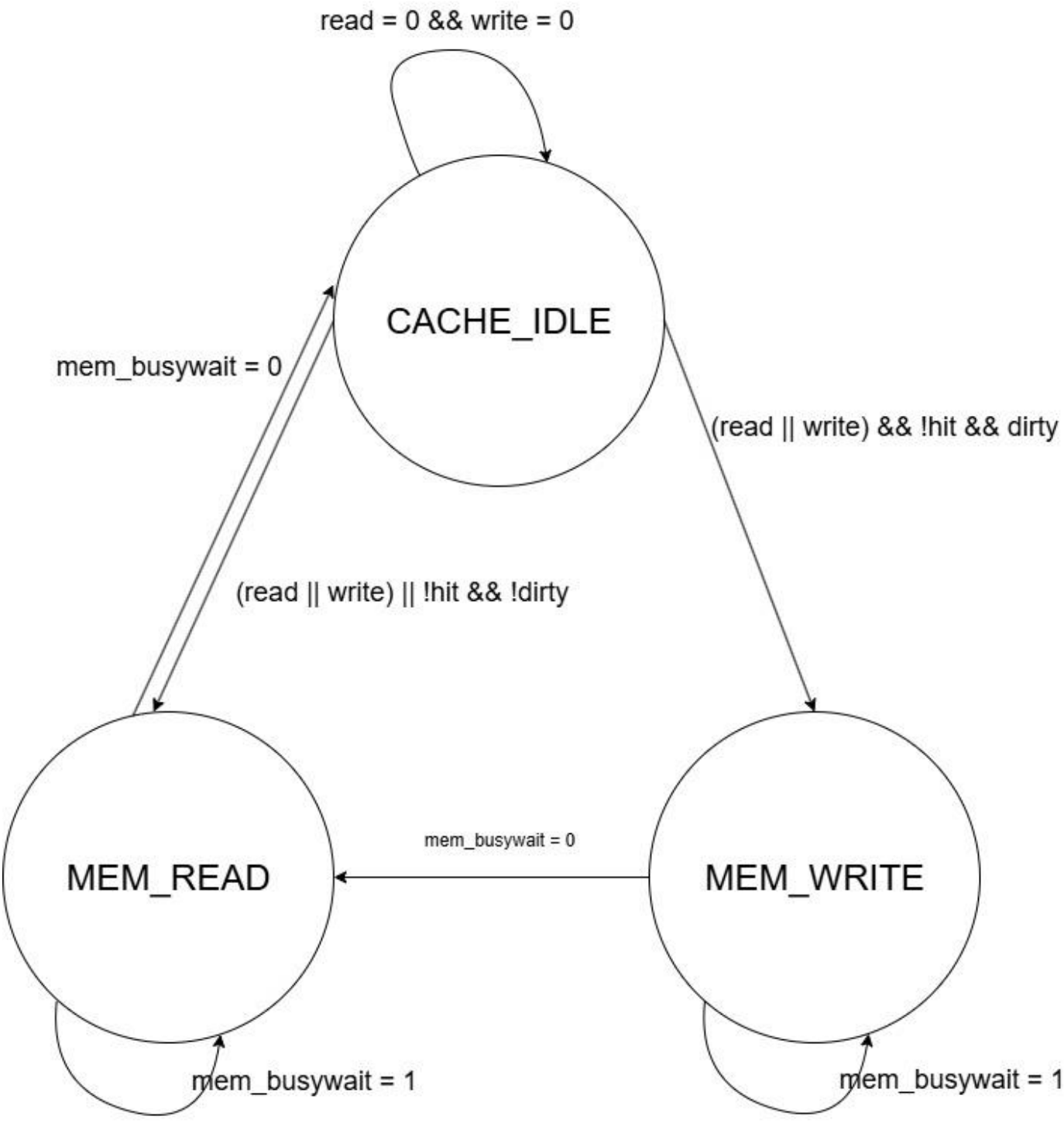
A.V. George – E/21/152

S. KAVISHANTHAN – E/21/220

Abstract

This report details the implementation and performance analysis of a data cache designed for the CO224 Lab 06 Part 2 by Group 09. The cache operates using a finite state machine with three states: IDLE, MEM_READ, and MEM_WRITE. We compare the cached implementation against a cache-less system across six cases, analyzing performance in terms of clock cycles for read and write operations under various hit and miss scenarios. The results demonstrate that the cache significantly improves performance when hit rates are high, leveraging temporal and spatial locality.

FSD of Cache Memory



1.1 State Descriptions

1. IDLE State:

• The cache remains idle when both read and write signals from the CPU are 0.

• Upon receiving a read or write signal, the address is split into tag, index, and offset after a 1 time unit delay.

• The cache block corresponding to the index is extracted, and the hit signal is asserted after a 0.9 time unit delay.

• Depending on hit, read, write, and dirty bit signals, the following actions occur:

Hit = 1, Read = 0, Write = 1: Data from WRITEDATA is written to the cache block at the next clock edge (write

hit).

Hit = 1, Read = 1, Write = 0: Data from the cache is sent to READDATA within one clock cycle (read hit).

Hit = 0, Read or Write = 1, Dirty = 0: Transition to MEM_READ state.

Hit = 0, Read or Write = 1, Dirty = 1: Transition to MEM_WRITE state.

2. MEM_READ State:

• Reads a 4-byte data block from memory based on the tag and index, taking 21 clock cycles (intended 20 cycles).

• After reading, the block is written to the cache with a 1 time unit delay.

• The cache returns to the IDLE state, reasserting the hit status.

3. MEM_WRITE State:

• If the dirty bit is 1, the cache block is written to memory, taking 21 clock cycles due to a race condition.

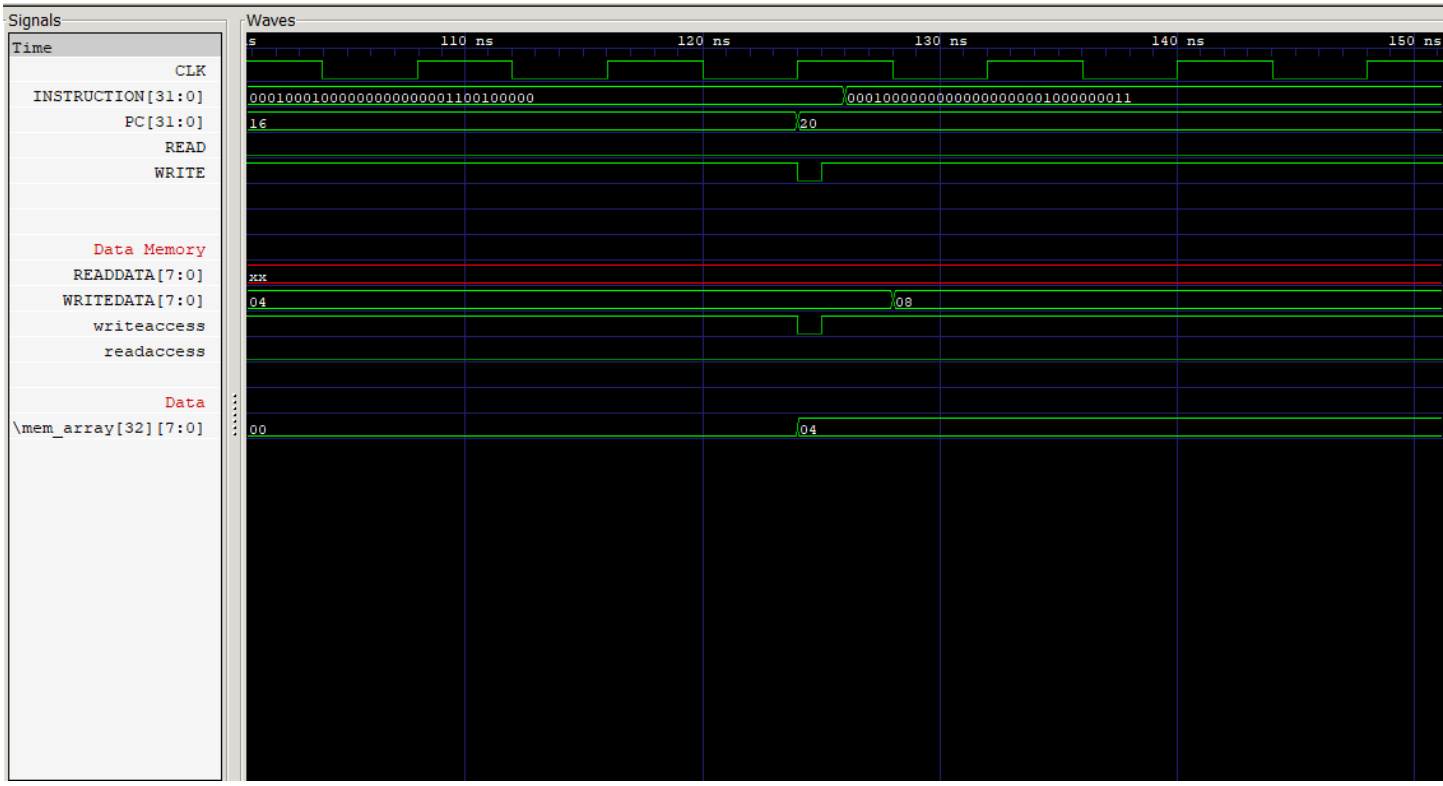
• Once the memory busywait signal is cleared, the state transitions to MEM_READ

Comparison

1) Write Miss, Dirty Bit = 0
   - Cached: A write miss with dirty bit = 0 requires fetching a 4-byte block from memory (21 cycles due to race condition), followed by writing data to the cache (1 cycle), totaling 22 cycles.
   - Cache-less: Writing 1 byte takes 5 cycles.

```
1   loadi 2 0x08      // Load immediate value 0x08 into register 2
2   loadi 3 0x04      // Load immediate value 0x04 into register 3
3   swi   2 0x10      // Store word immediate: store value in reg 2 to memory addr 0x10
4   add   0 2 3       // Add contents of reg 2 and 3, store result in reg 0
5   swi   3 0x20      // Store word immediate: store value in reg 3 to addr 0x20 (e.g., cache miss, dirty=0)
6   swd   2 3         // Store word from reg 2 to memory address in reg 3
```

Without Cache



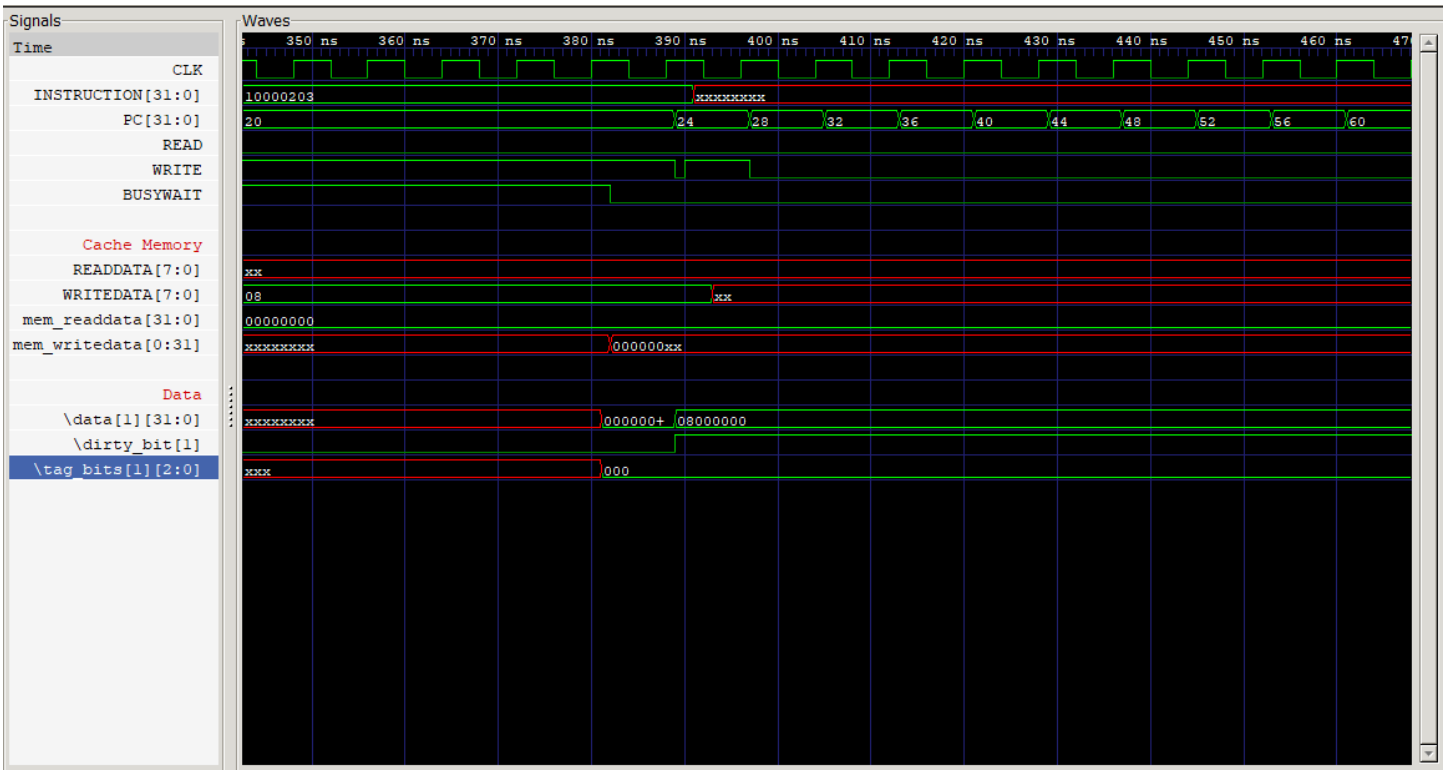
CHANGE OF REGISTER VALUES

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	8	0	0	0	0	0
21	0	0	8	4	0	0	0	0
77	12	0	8	4	0	0	0	0

With Cache

CHANGE OF REGISTER VALUES

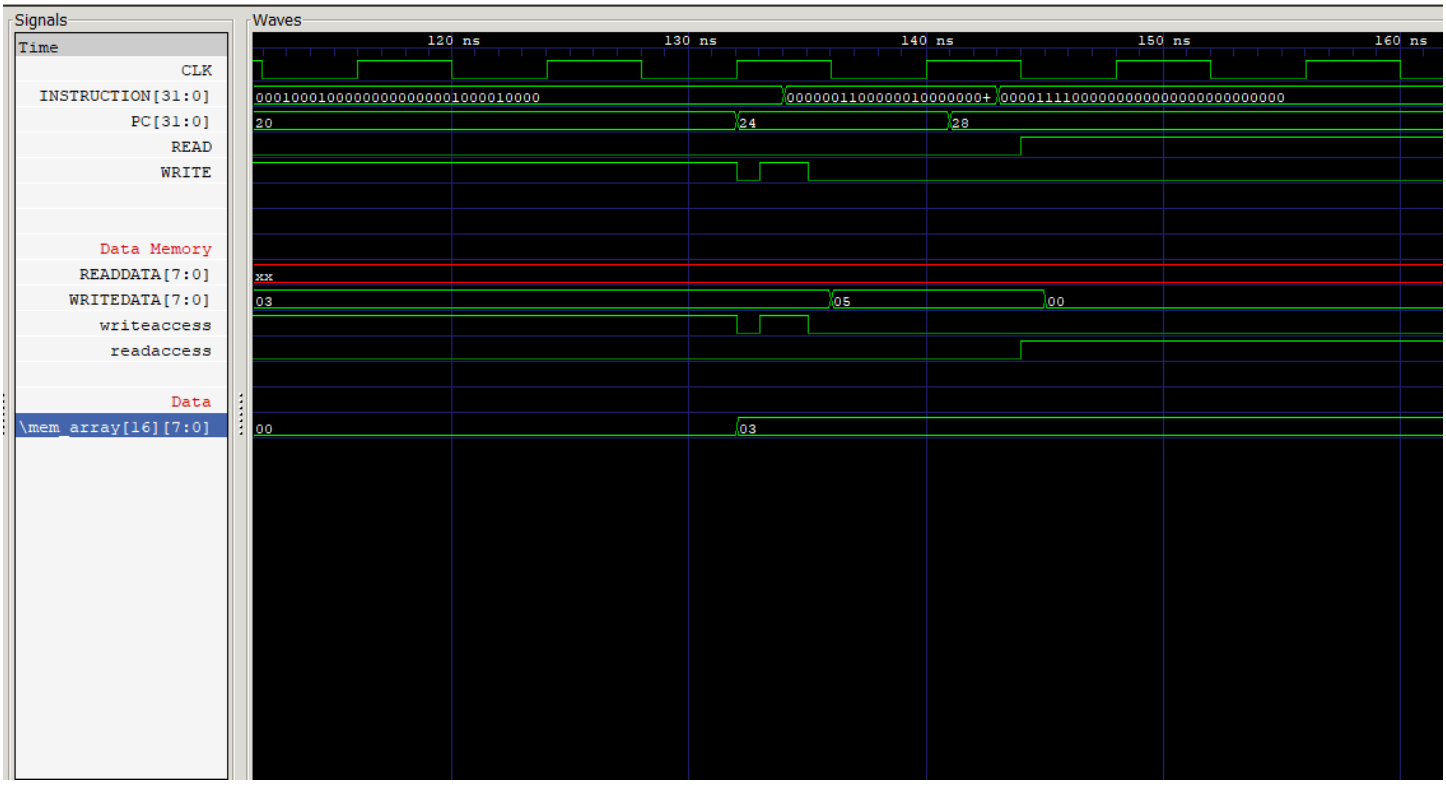
time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	8	0	0	0	0	0
21	0	0	8	4	0	0	0	0
205	12	0	8	4	0	0	0	0



2) Write Miss, Dirty Bit = 1
   - Cached: A write miss with dirty bit = 1 requires writing the cache block to memory (21 cycles), fetching a new block (21 cycles), and writing data (1 cycle), totaling 42 cycles.
   - Cache-less: Writing 1 byte takes 5 cycles.

```
loadi 1 0x02       // Load 0x02 into register 1
loadi 2 0x03       // Load 0x03 into register 2
loadi 3 0x04       // Load 0x04 into register 3
swi   1 0x00       // Store register 1 to memory address 0x00
add   1 1 2        // Add reg 1 and 2, store result in reg 1
swi   2 0x10       // Store reg 2 to memory address 0x10 (cache miss, write=1, dirty=1)
sub   2 1 2        // Subtract reg 2 from reg 1, store in reg 2
lwi   0 0x00       // Load word from memory address 0x00 into reg 0
```

Without Cache:



CHANGE OF REGISTER VALUES

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	2	0	0	0	0	0	0
21	0	2	3	0	0	0	0	0
29	0	2	3	4	0	0	0	0
85	0	5	3	4	0	0	0	0
141	0	5	2	4	0	0	0	0
189	2	5	2	4	0	0	0	0

With Cache:

CHANGE OF REGISTER VALUES

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	2	0	0	0	0	0	0
21	0	2	3	0	0	0	0	0
29	0	2	3	4	0	0	0	0
213	0	5	3	4	0	0	0	0
397	0	5	2	4	0	0	0	0
405	x	5	2	4	0	0	0	0

3) Write Hit

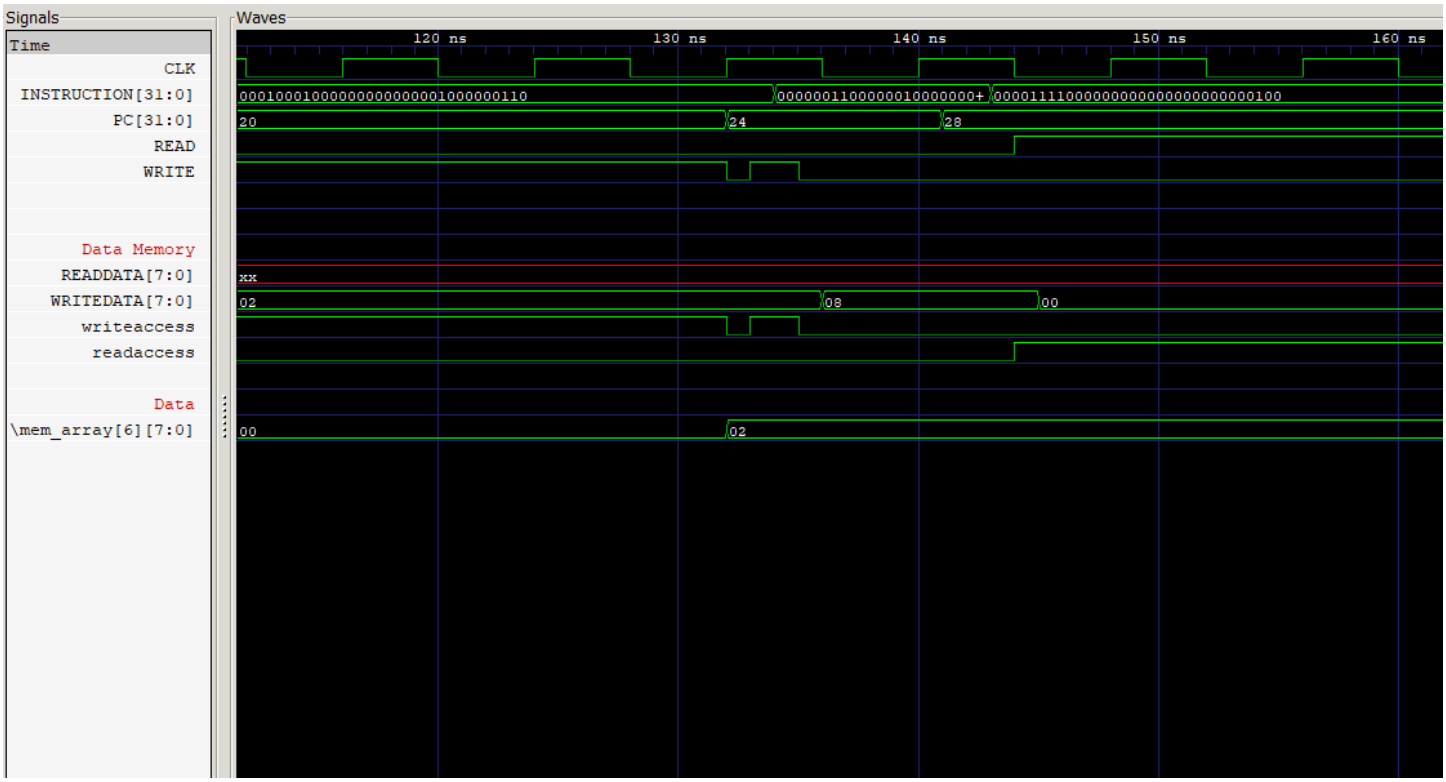
- Cached: A write hit writes data to the cache in 1 cycle, with the hit signal asserted 2 time units after address receipt.
- Cache-less: Writing 1 byte takes 5 cycles.

```
loadi 1 0x06       // Load 0x06 into register 1
loadi 2 0x02       // Load 0x02 into register 2
loadi 3 0x08       // Load 0x08 into register 3
swi   1 0x04       // Store register 1 to memory address 0x04
add   1 1 2        // Add reg 1 and 2, store result in reg 1
swi   2 0x06       // Store reg 2 to memory address 0x06 (cache hit, write = 1)
sub   2 1 2        // Subtract reg 2 from reg 1, store result in reg 2
lwi   0 0x04       // Load word from memory address 0x04 into reg 0
```

Without Cache:

CHANGE OF REGISTER VALUES

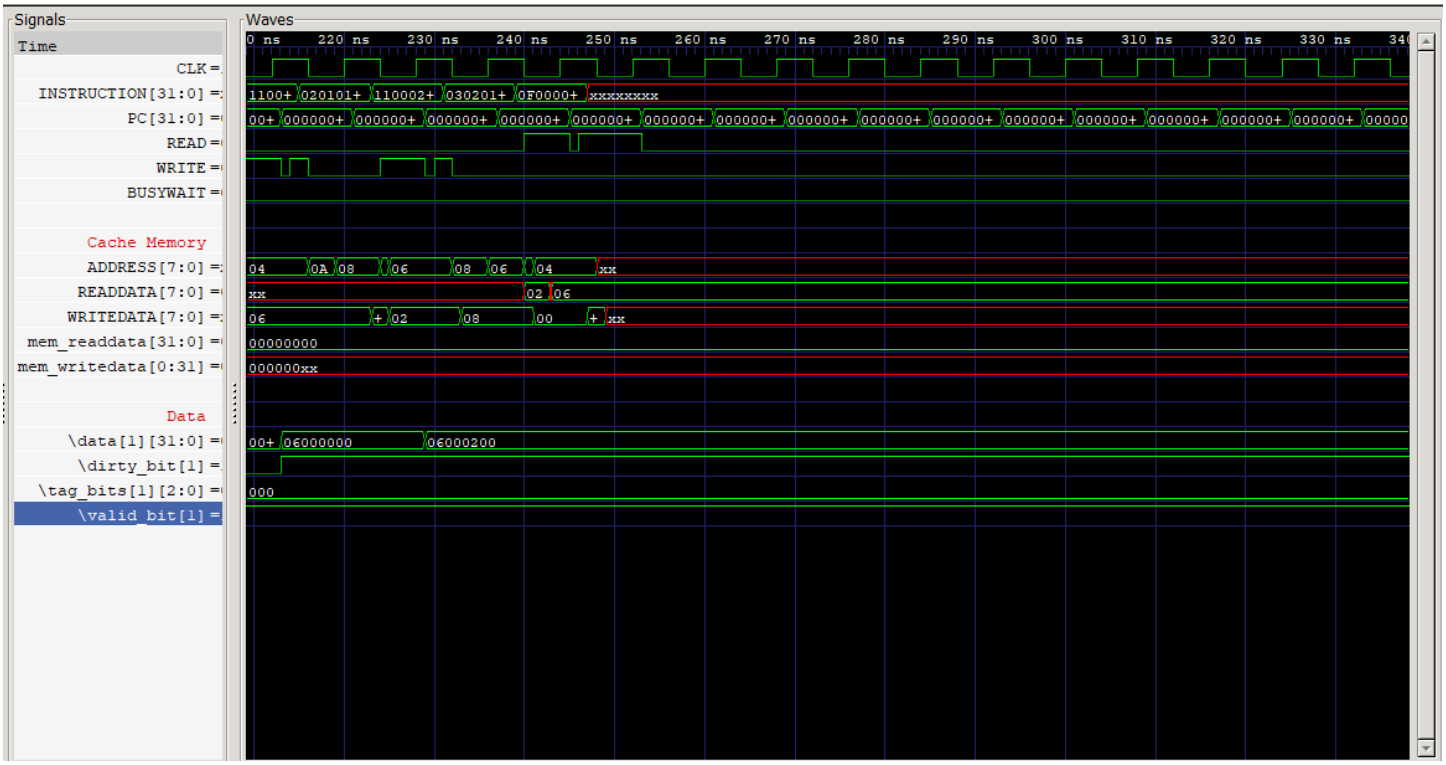
time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	6	0	0	0	0	0	0
21	0	6	2	0	0	0	0	0
29	0	6	2	8	0	0	0	0
85	0	8	2	8	0	0	0	0
141	0	8	6	8	0	0	0	0
189	6	8	6	8	0	0	0	0



With Cache:

CHANGE OF REGISTER VALUES

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	6	0	0	0	0	0	0
21	0	6	2	0	0	0	0	0
29	0	6	2	8	0	0	0	0
221	0	8	2	8	0	0	0	0
237	0	8	6	8	0	0	0	0
245	6	8	6	8	0	0	0	0



4) Read Miss, Dirty Bit = 0

• Cached: A read miss with dirty bit = 0 requires fetching a 4-byte block from memory (21 cycles), followed by reading data to the CPU (1 cycle), totaling 22 cycles.

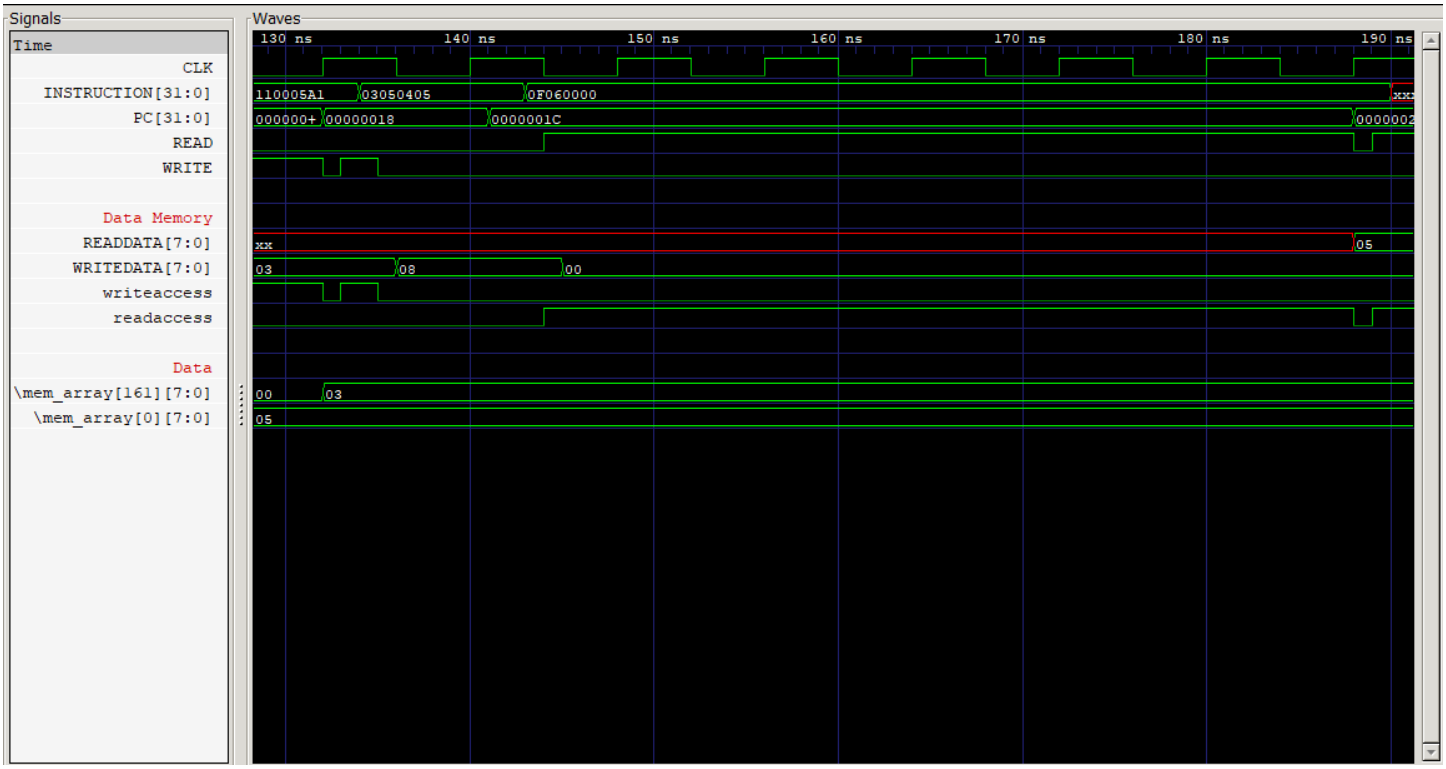
• Cache-less: Reading 1 byte takes 5 cycles.

```
1   loadi 4 0x05        // R4 = 0x05
2   loadi 5 0x03        // R5 = 0x03
3   loadi 7 0x07        // R7 = 0x07
4   swi   4 0x00        // Store R4 to memory[0x00] → block loaded, clean
5   add   4 4 5         // R4 = 0x05 + 0x03 = 0x08 (no write to 0x00)
6   swi   5 0xA1        // Store R5 to address 0xA1 (different block)
7   sub   5 4 5         // R5 = 0x08 - 0x03 = 0x05 (again, not touching 0x00)
8   lwi   6 0x00        // Load from 0x00 → read miss, but dirty bit = 0
```

Without Cache

CHANGE OF REGISTER VALUES

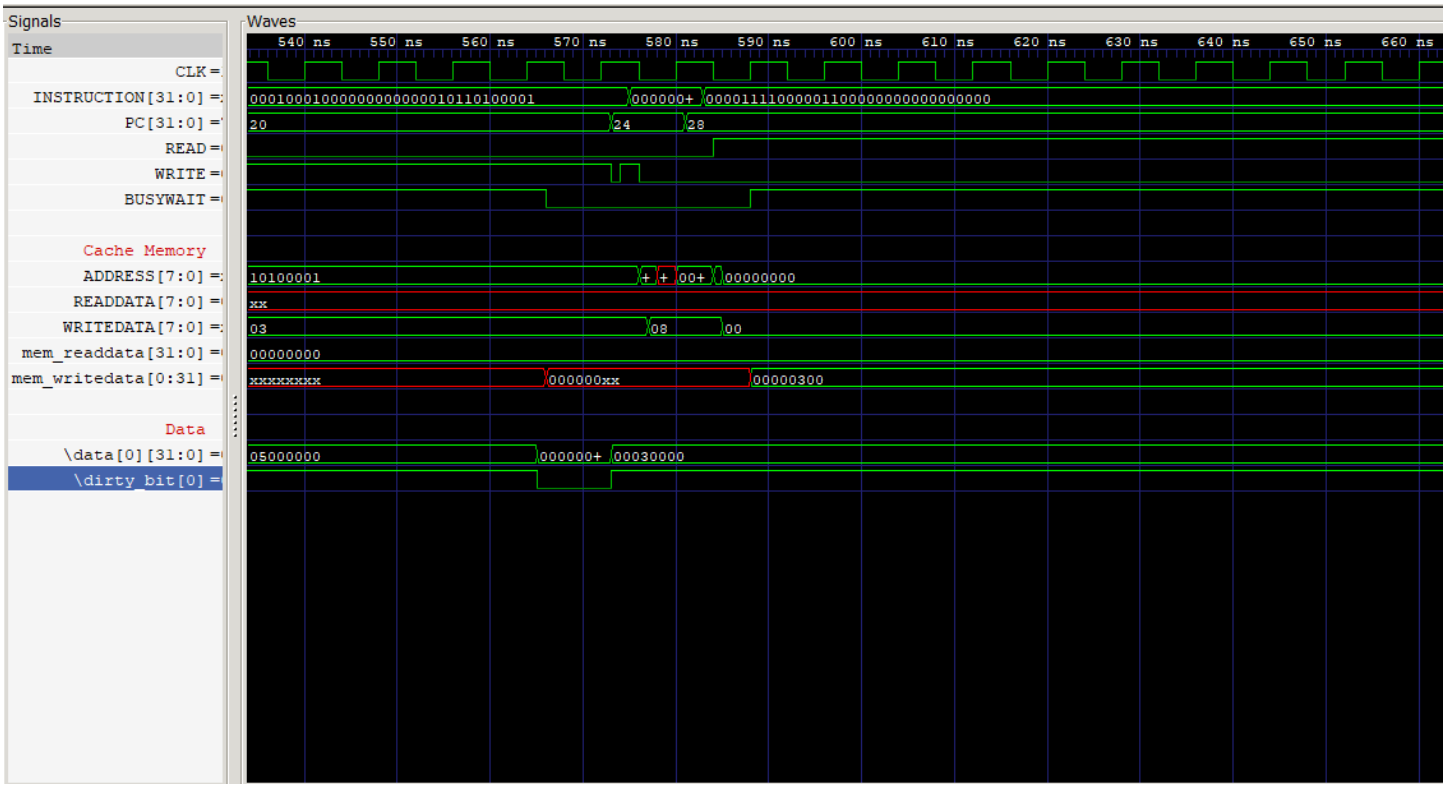
time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
85	0	0	0	0	8	3	0	7
141	0	0	0	0	8	5	0	7
189	0	0	0	0	8	5	5	7



With Cache

CHANGE OF REGISTER VALUES

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	0	0	5	0	0	0
21	0	0	0	0	5	3	0	0
29	0	0	0	0	5	3	0	7
221	0	0	0	0	8	3	0	7
581	0	0	0	0	8	5	0	7
933	0	0	0	0	8	5	5	7



5) Read Miss, Dirty Bit = 1

• Cached: A read miss with dirty bit = 1 requires writing the cache block to memory (21 cycles), fetching a new block (21 cycles), and reading data (1 cycle), totaling 42 cycles.

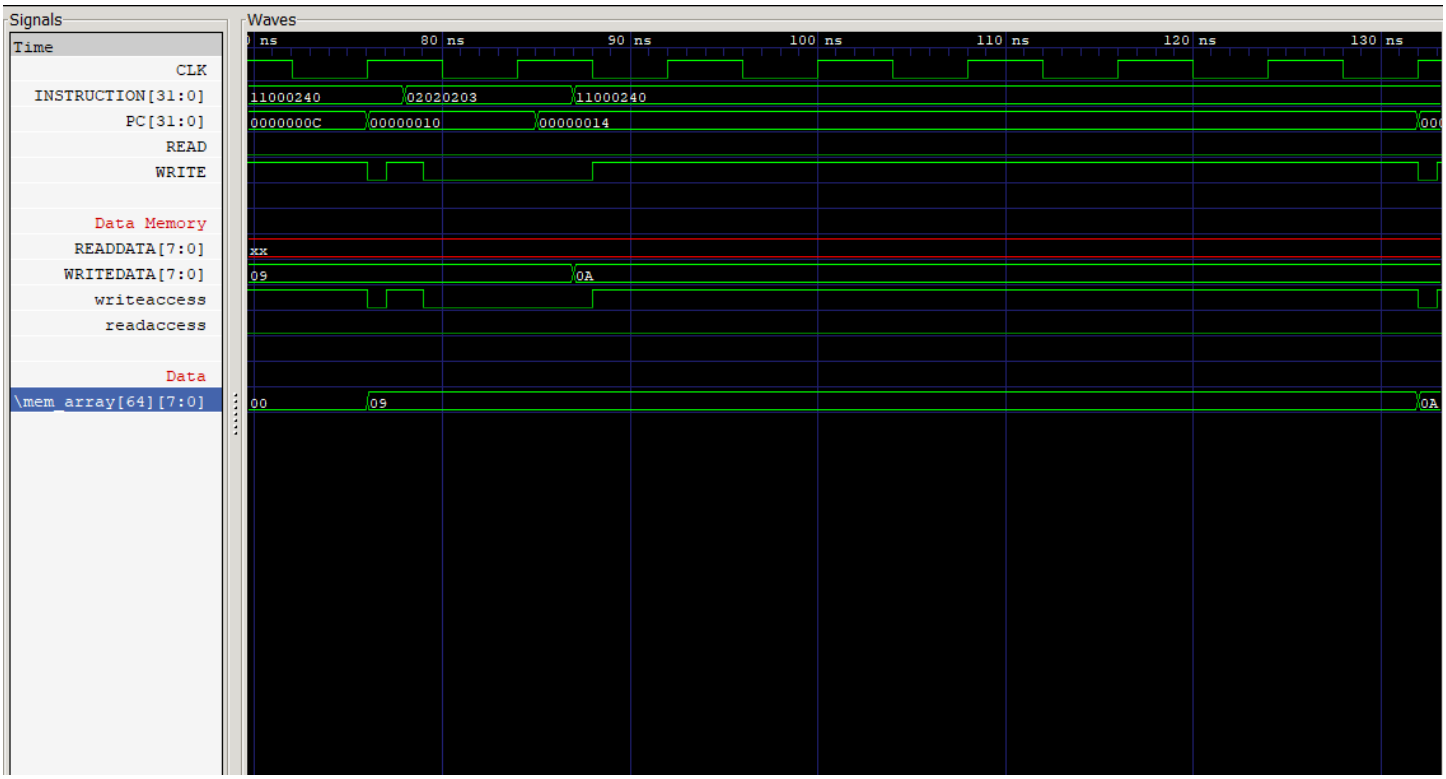
• Cache-less: Reading 1 byte takes 5 cycles.

```
loadi 2 0x09        // R2 = 0x09
loadi 3 0x01        // R3 = 0x01
loadi 7 0x04        // R7 = 0x04

swi   2 0x40        // Store R2 to address 0x40 → block loaded, clean

add   2 2 3         // R2 = R2 + R3 = 0x0A

swi   2 0x40        // Store new R2 to same address → block marked dirty

swi   7 0xA4        // Unrelated store

sub   3 2 3         // R3 = R2 - R3 = 0x0A - 0x01 = 0x09

lwi   6 0x40        // Load from 0x40 → cache miss, dirty = 1 → triggers write-back
```

Without Cache

CHANGE OF REGISTER VALUES

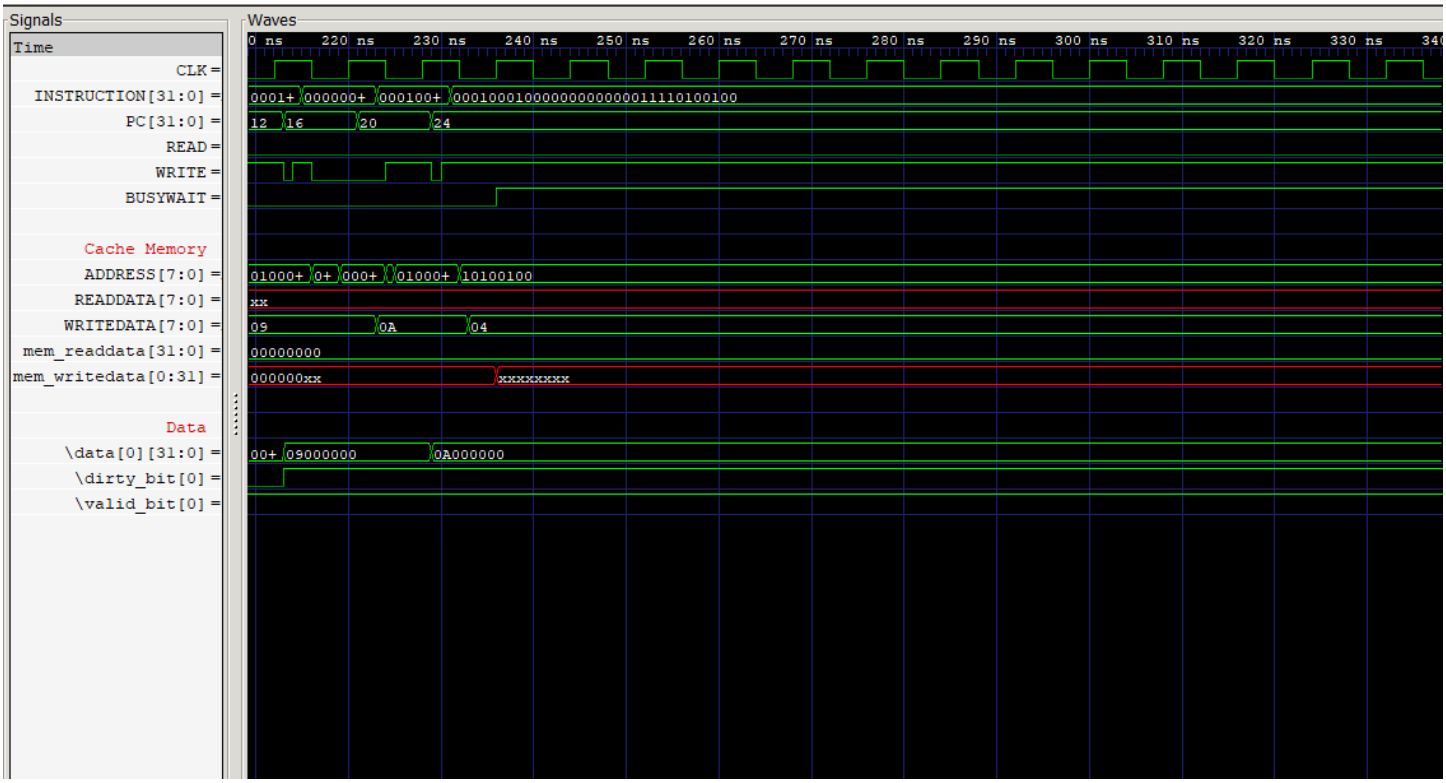
time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	9	0	0	0	0	0
21	0	0	9	1	0	0	0	0
29	0	0	9	1	0	0	0	4
85	0	0	10	1	0	0	0	4
189	0	0	10	9	0	0	0	4
237	0	0	10	9	0	0	10	4



With Cache

CHANGE OF REGISTER VALUES

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	0	9	0	0	0	0	0
21	0	0	9	1	0	0	0	0
29	0	0	9	1	0	0	0	4
221	0	0	10	1	0	0	0	4
421	0	0	10	9	0	0	0	4
429	0	0	10	9	0	0	10	4



6) Read Hit

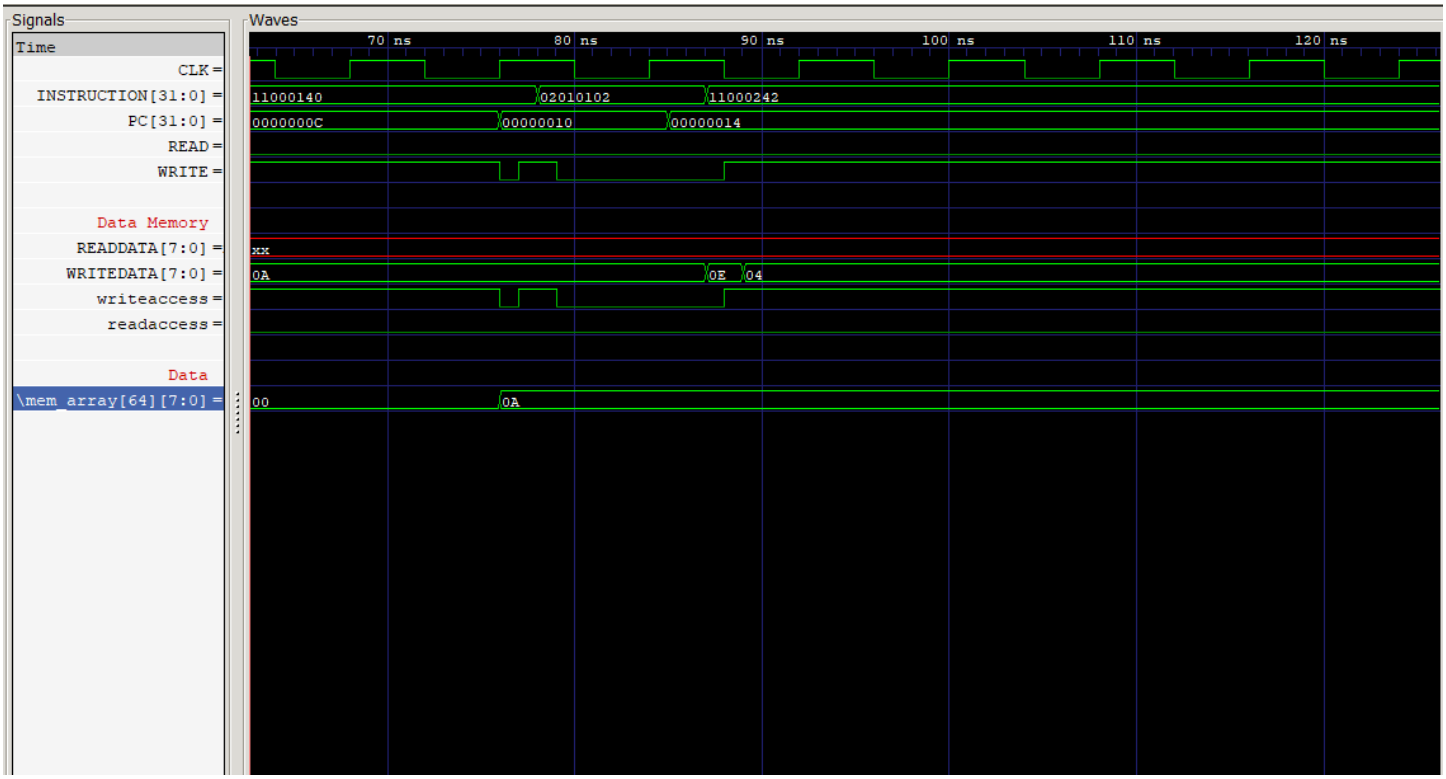
• Cached: A read hit reads data from the cache in 1 cycle, with the hit signal asserted 2 time units after address receipt.

• Cache-less: Reading 1 byte takes 5 cycles.

Without Cache

CHANGE OF REGISTER VALUES

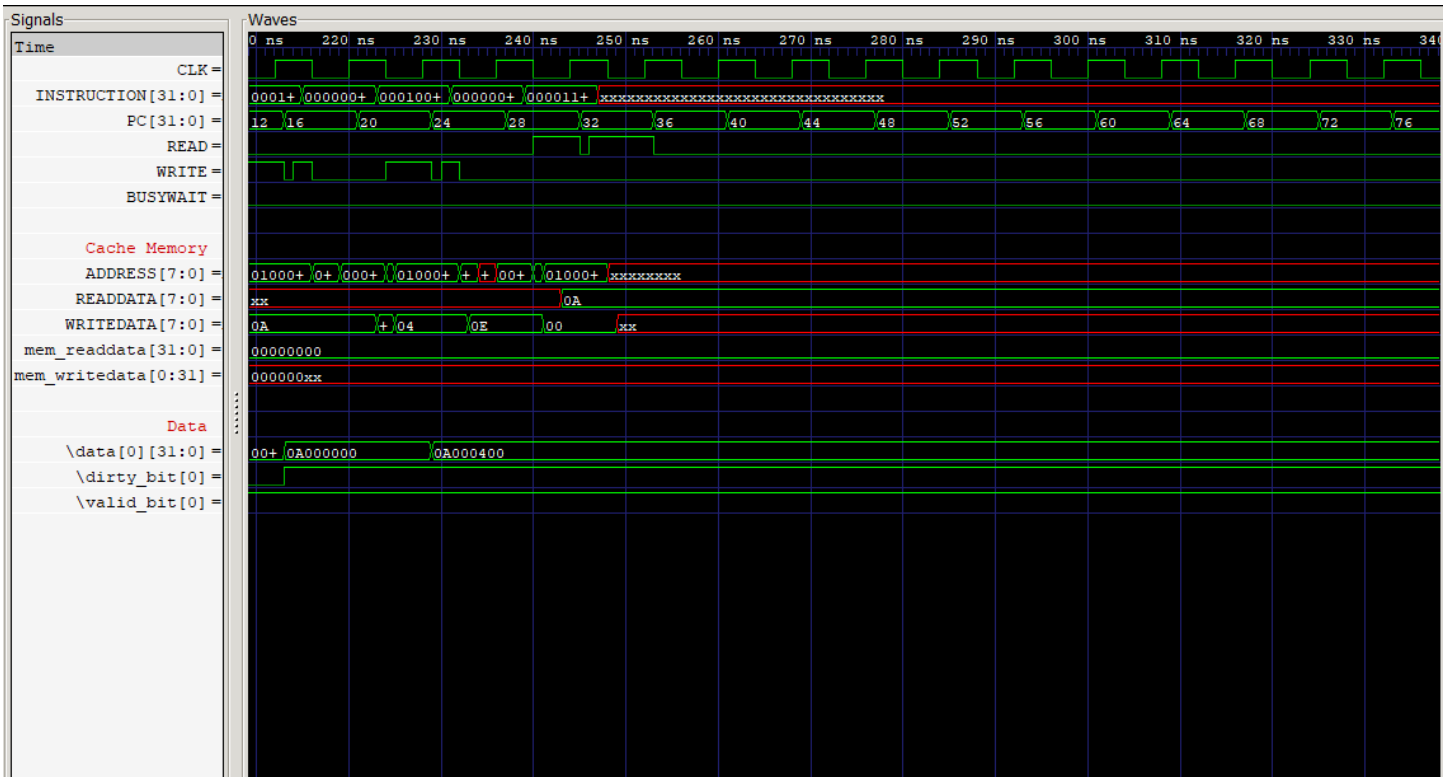
time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	10	0	0	0	0	0	0
21	0	10	4	0	0	0	0	0
29	0	10	4	0	0	1	0	0
85	0	14	4	0	0	1	0	0
141	0	14	10	0	0	1	0	0
189	0	14	10	10	0	1	0	0



With Cache

CHANGE OF REGISTER VALUES

time	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
5	0	0	0	0	0	0	0	0
13	0	10	0	0	0	0	0	0
21	0	10	4	0	0	0	0	0
29	0	10	4	0	0	1	0	0
221	0	14	4	0	0	1	0	0
237	0	14	10	0	0	1	0	0
245	0	14	10	10	0	1	0	0



The cache implementation outperforms the cache-less system when cache hits dominate, leveraging temporal and spatial locality. Read and write hits complete in 1 clock cycle, compared to 5 cycles in the cache-less system. However, cache misses, especially with a dirty bit set, incur significant overhead (up to 42 cycles). High hit rates are critical for the cache to reduce program execution time effectively.