CO224 –LAB 06

PART 2 – DATA CACHE

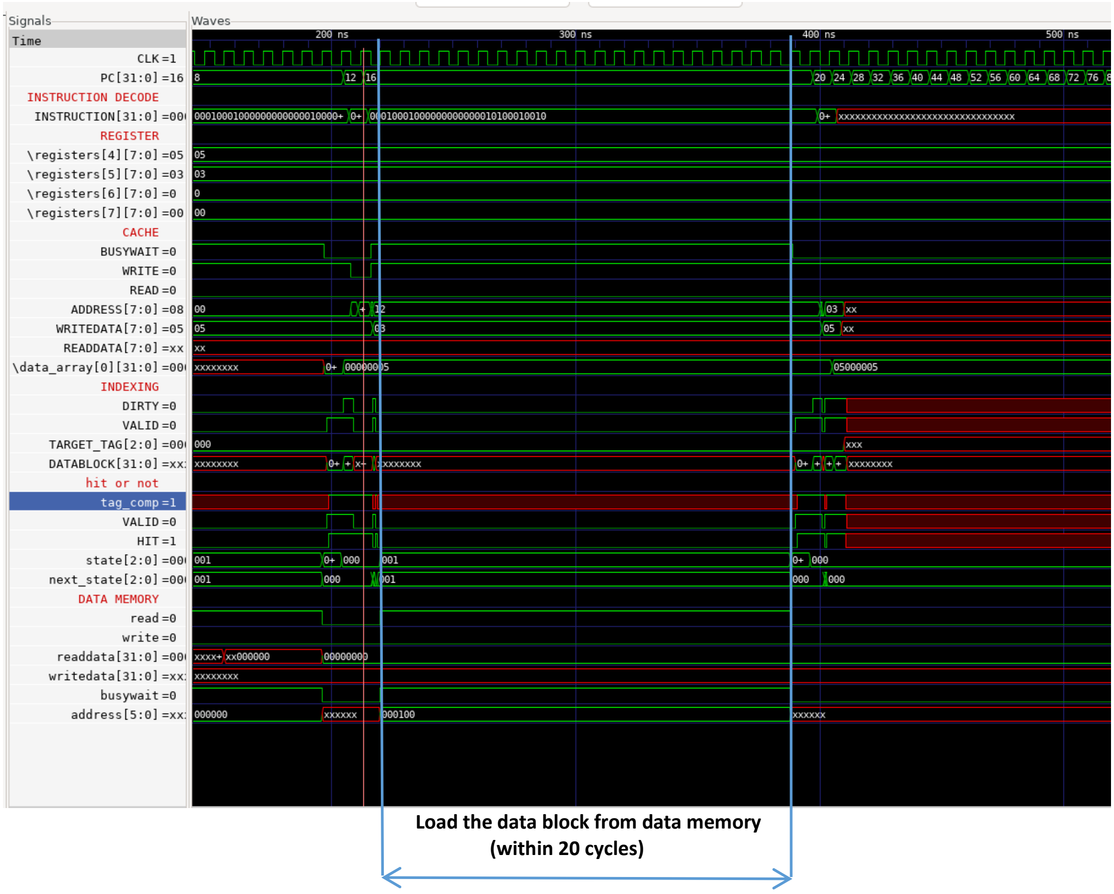
GROUP 06
DEWAGEDARA D.M.E.S. (E/21/087)
PERERA W.S.S. (E/21/302)

Case 1 - Cache miss occurs when write =1 and read = 0 (write miss) and the dirty bit =0

Code

```
loadi 4 0x05
loadi 5 0x03
swi 4 0x00 //write miss with dirty = 0
add 1 4 5
swi 5 0x12  //write miss with dirty = 0
swd 4 5
```

Timing diagram - WITH CACHE



Here, the instruction **`swi 5 0x12`** causes a write miss because address 0x12 is not yet loaded into the cache.

Since the dirty bit = 0, this means:

- The existing cache block (in that index) is clean.
- So no write-back to memory is needed before loading the new block.

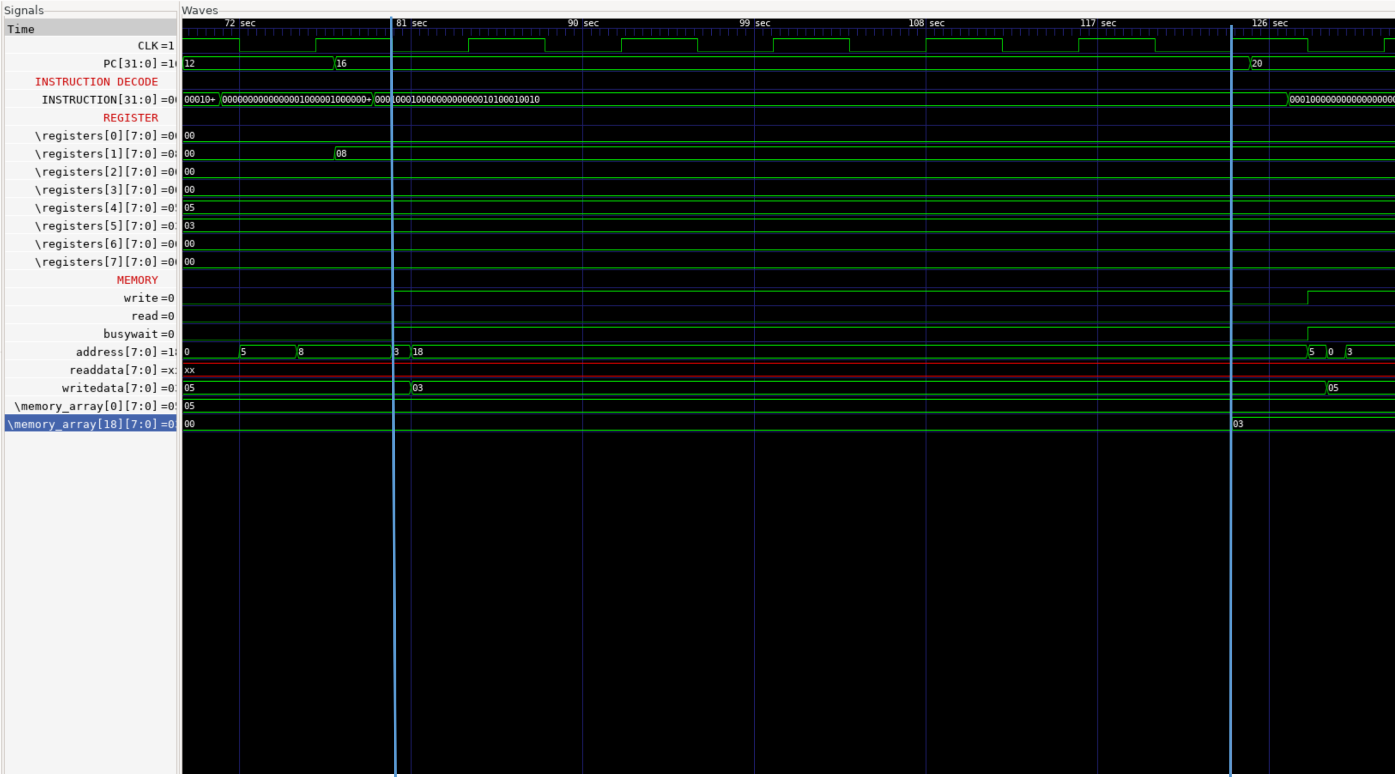
However, the cache must still load the entire block (e.g., 4 words = 16 bytes) from main memory before writing.

If:

- One word memory access = 5 cycles
- One block = 4 words
- Then block load = 5 × 4 = 20 cycles
- Then the write takes another 20 cycles after loading.

Total = 20 cycles for this write with cache.

Timing diagram - WITHOUT CACHE



Directly write in data memory (Within 5 clock cycles)

If the cache is not used, the **`swi 5 0x12`** directly writes the value in register R5 to memory address 0x12. There is:

- No block fetching
- No tag checking
- No dirty/valid bit checking

So the write happens in one memory access = 5 cycles.

Therefore, making it significantly faster than the cached version in this specific scenario.

NOTE : This Is Not a Typical "Write Miss with Dirty = 0"

While this situation behaves like a write miss with dirty = 0, it is technically a **cold miss** because:

- The cache is initially invalid (valid = 0), as no data has been loaded into that index yet.
- The dirty bit is 0 by default, not because of a prior clean write.

Conflict Misses Also Behave the Same

This exact delay also occurs during a **conflict miss** when:

- A valid, clean block is replaced by a new block mapping to the same index.
- Since the dirty bit is 0, no write-back occurs.
- But the cache still loads the entire new block before writing.

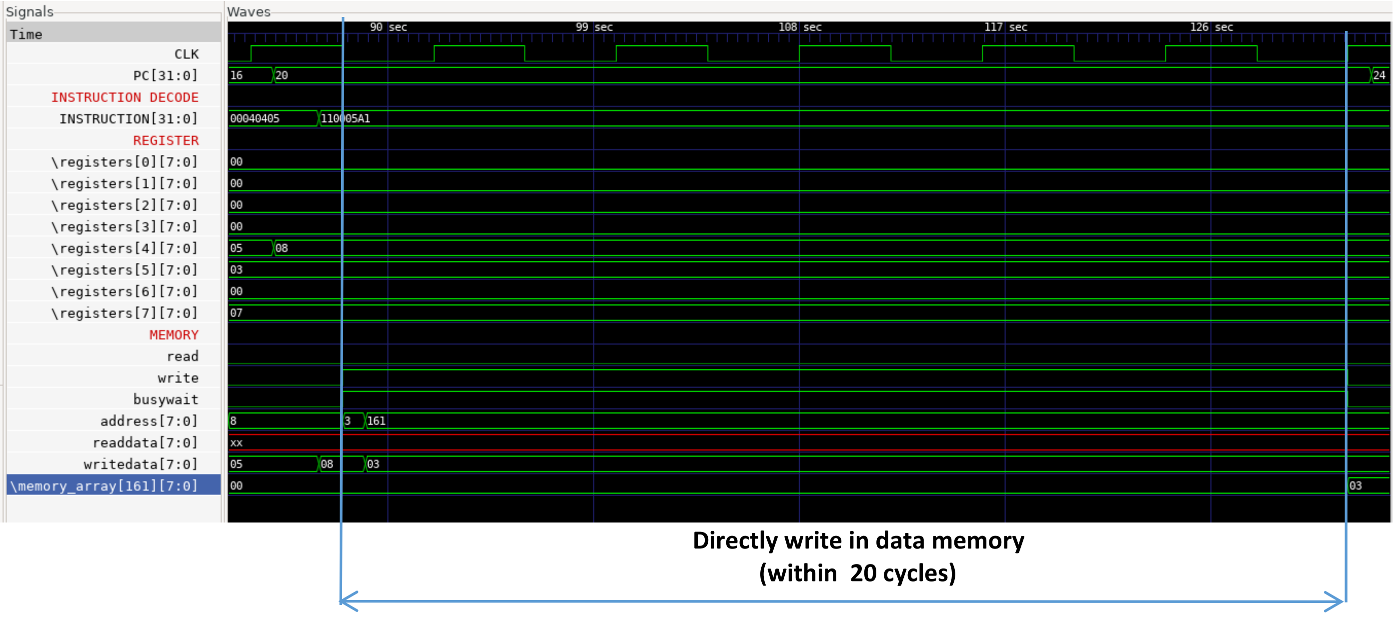
So, a write miss due to conflict (dirty = 0) also get the same delay.

Case 2 - Cache miss occurs when write =1 and read = 0 (write miss) and the dirty bit=1(write back)

Code

```
loadi 4 0x05
loadi 5 0x03
loadi 7 0x07
swi 4 0x00
add 4 4 5
swi 5 0xA1     //Write miss with dirty = 1
```

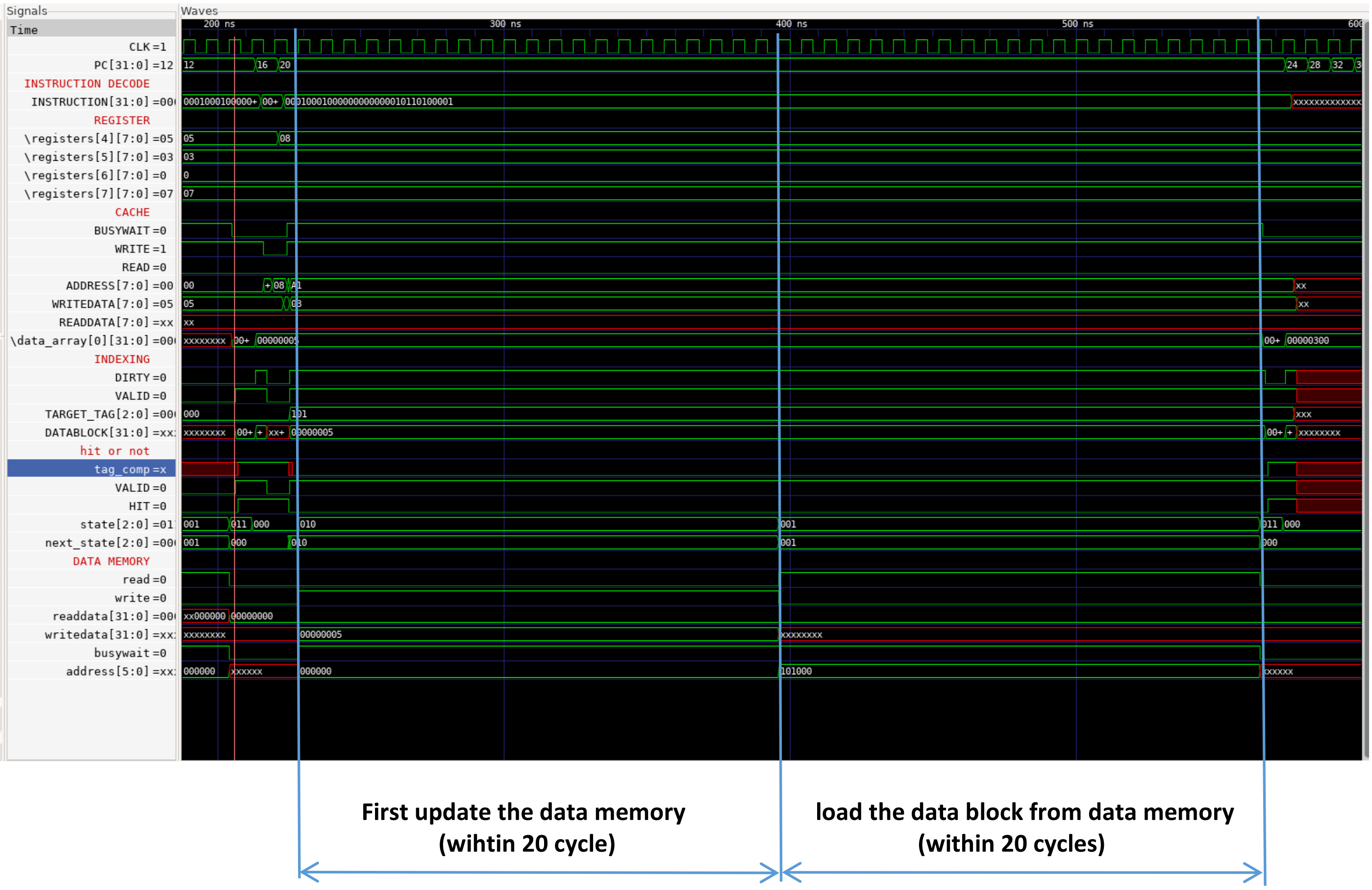
Timing diagram - WITHOUT CACHE



In a no-cache system, the **`swi 5 0xA1`** :

- Directly writes the value 0x03 to memory address 0xA1.
- No need to write back or fetch any blocks.
- Only takes 5 cycles.

Timing diagram - WITH CACHE



When **`swi 5 0xA1`** is executed:

- Address 0xA1 maps to a different block than 0x00 (same index in direct-mapped cache).
- The currently cached block (e.g., containing 0x00) is:
  - Valid = 1
  - Dirty = 1

This is a write-back scenario:

- The dirty block (ex: from 0x00) must first be written back to main memory (4 words × 5 cycles = 20 cycles).
- The new block containing 0xA1 must then be loaded from memory (another 20 cycles).

Finally, the write to 0xA1.

For update the cache,

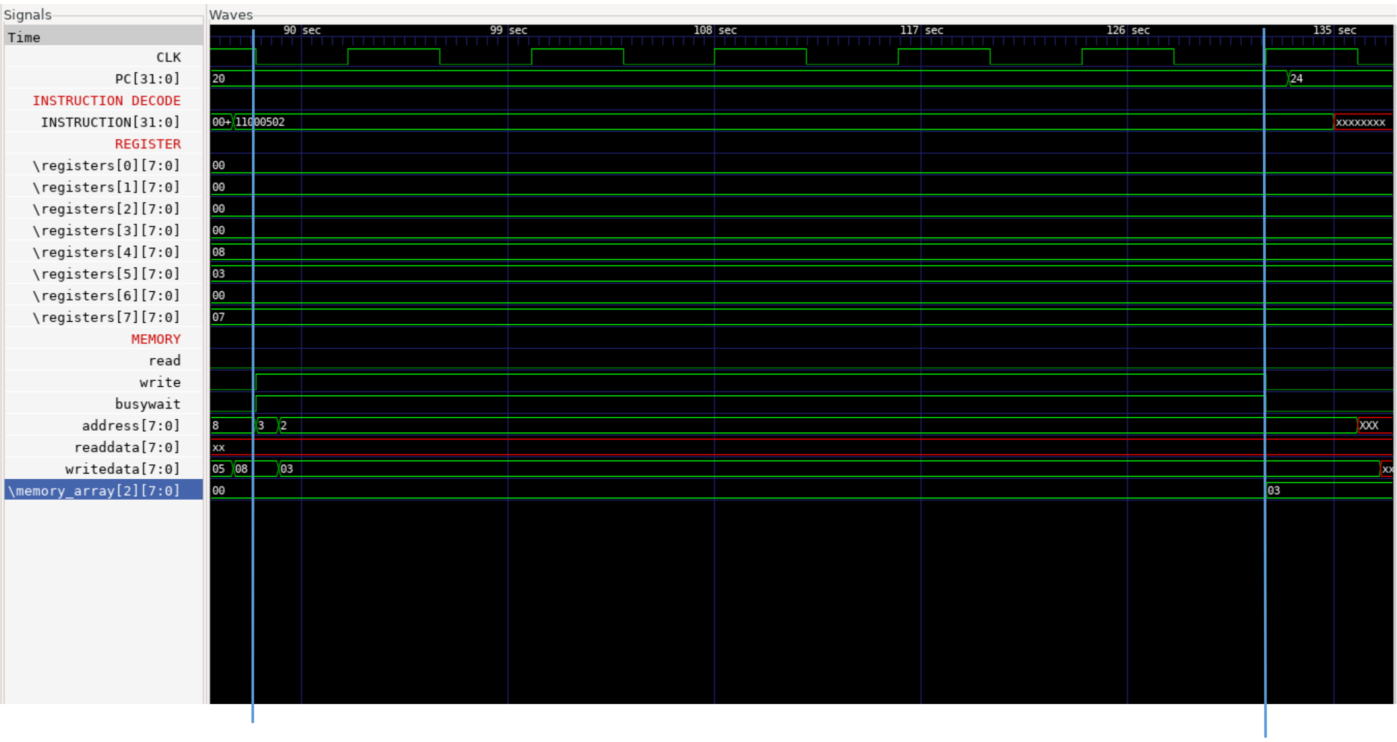
Total time = 20 (write-back) + 20 (load)
= 40 cycles

Case 3: Cache hit when write = 1 and read = 0 (write hit)

Code

```
loadi 4 0x05
loadi 5 0x03
loadi 7 0x07
swi 4 0x00
add 4 4 5
swi 5 0x02          //Write hit
```

Timing diagram - WITHOUT CACHE

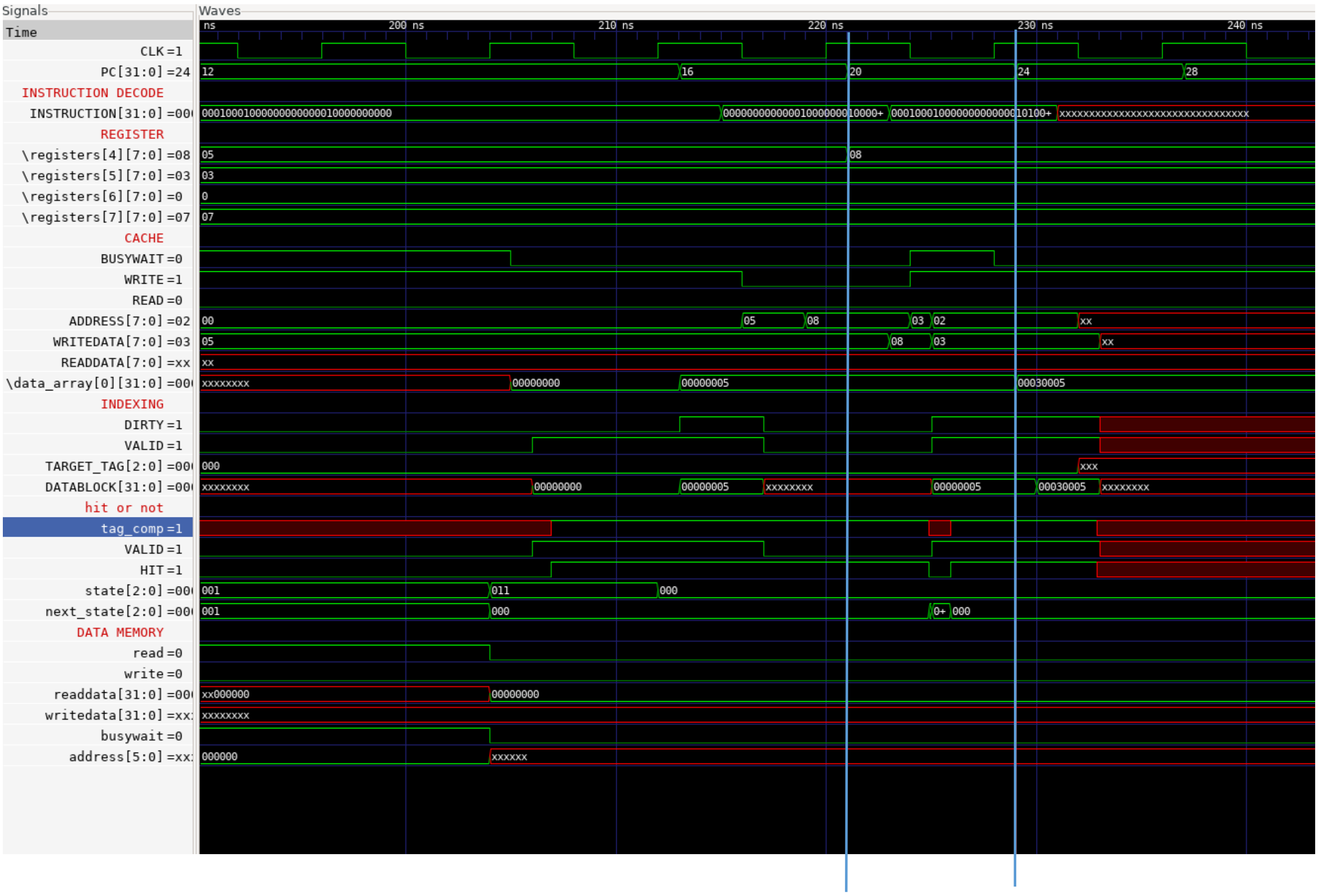


Every store accesses the main memory directly.

- Writing to memory address 0x02 takes:
- 1 memory access = 5 cycles

Total Delay = 40 cycles

Timing diagram - WITH CACHE



swi 4 0x00 loads the block containing 0x00 (e.g., 0x00–0x03) into the cache.

- Block is marked valid, and after write, dirty = 1.

Then, **swi 5 0x02** stores to address 0x02:

- This is in the same block, so the data is already in the cache.
- This is a write hit.

Timing:

- On a write hit, the cache updates the word within the cache block.
- This update happens within 1-cycle latency.
- The memory is not accessed during this write (since it's deferred in write-back caches).

Total Delay = 1 cycle (1 unit delay for write).

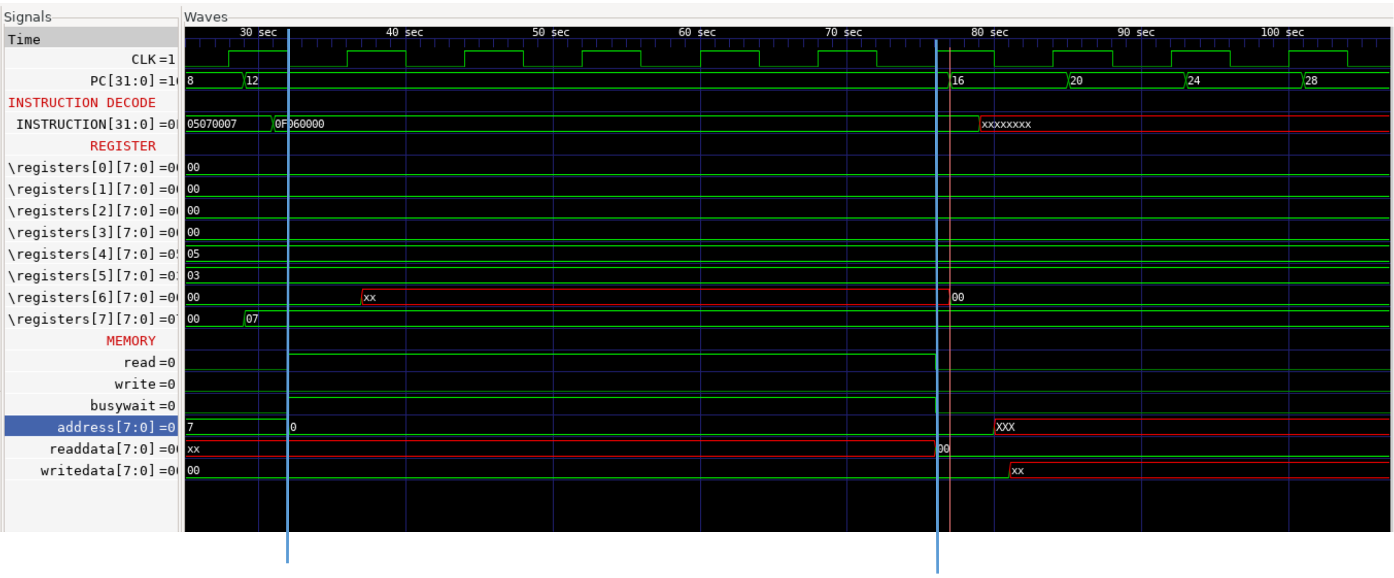
This comparison highlights the performance advantage of cache on write hits, where cache systems significantly reduce memory latency by performing local updates.

Case 4:Cache miss occurs when read =1 and write = 0 (read miss) and the dirty bit =0

Code,

```
part6_1 > SAMPLEPROGRAM.s
loadi 4 0x05
loadi 5 0x03
loadi 7 0x07
lwi 6 0x00       //Read miss when dirty bit = 0;
```

Timing diagram - WITHOUT CACHE

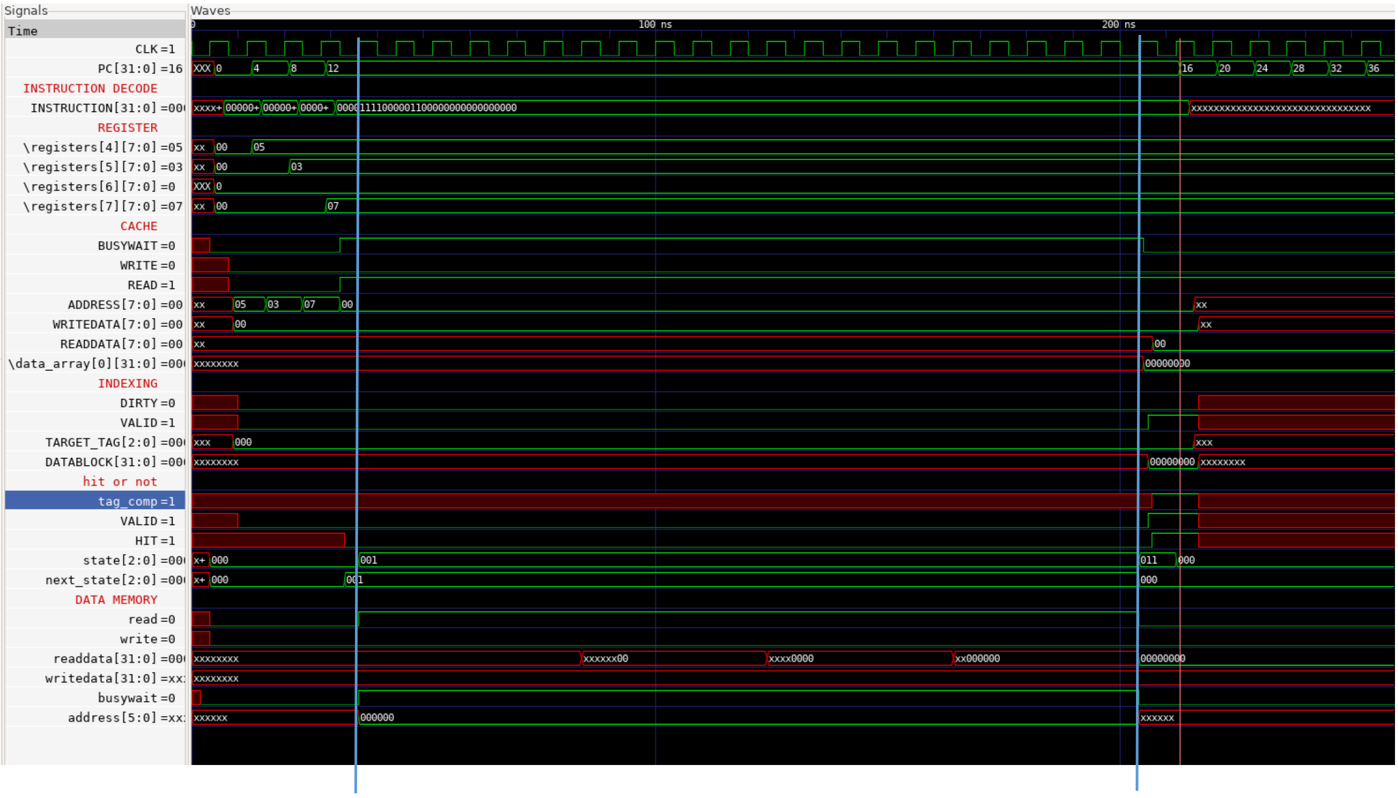


If the cache is disabled:

- The load instruction **lwi 6 0x00** directly accesses memory
- No block loading, just one word

Total Delay = 5 cycles

Timing diagram - WITH CACHE



With Cache (Read Miss, Dirty = 0)

- **`lwi 6 0x00`** reads from memory address 0x00.
- At this point, the cache is empty (first read), so:
  - Valid bit = 0
  - Dirty bit = 0 (by default)

This is a read miss, and more specifically, a **cold miss** — the first time this address or block is being accessed.
What Happens:

Since the block is invalid and not dirty, there is:

- No need to write back any existing data
- The cache simply fetches the entire block containing address 0x00 from memory.

Timing:

Assuming,

- One memory word = 5 cycles
- Block size = 4 words (16 bytes)

Then,

- Fetching the block from memory = 5 × 4 = 20 cycles

The actual read from cache after load = negligible, happens immediately

Total Delay = 20 cycles

Same Timing for Conflict Miss (Dirty = 0)

Suppose the cache already had a valid, clean block at the index for 0x00, but a new address mapping to the same index replaces it (a **conflict miss** in a direct-mapped cache):

- Old block is valid and clean (dirty = 0) → no write-back
- New block must be fetched from memory
- Same process as cold miss

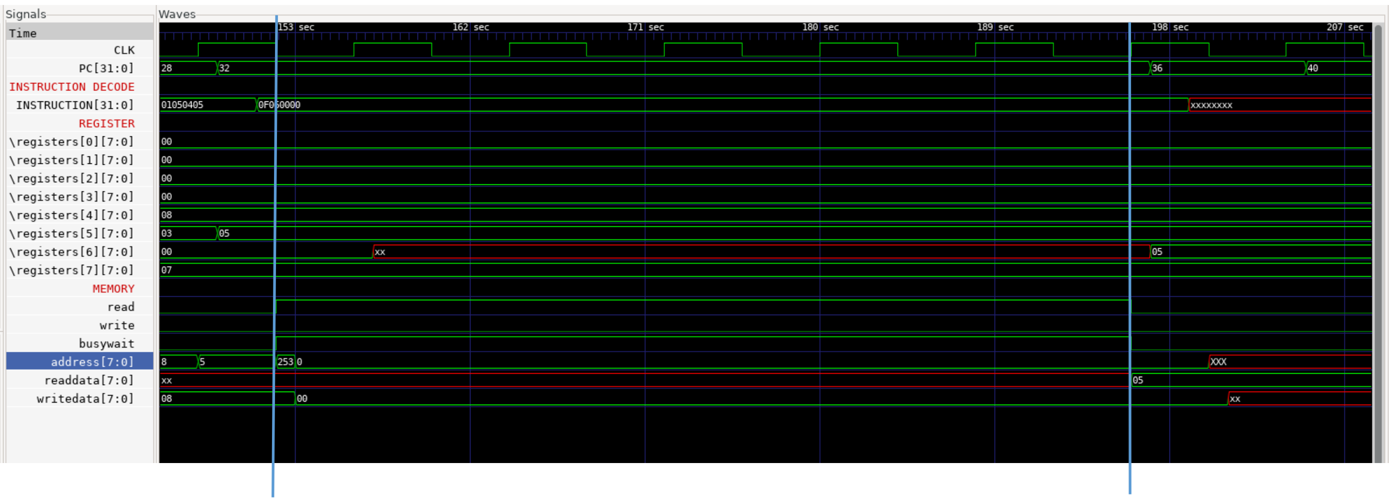
Total Delay = 20 cycles (identical to cold miss)

Case 5:Cache miss occurs when read =1 and write = 0 (read miss) and the dirty bit =1 write back)

<u>Code</u>

```
loadi 4 0x05
loadi 5 0x03
loadi 7 0x07
swi 4 0x00
add 4 4 5
swi 5 0xA1
swi 7 0xA2
sub 5 4 5
lwi 6 0x00   //Cache miss when dirty bit = 1;
```

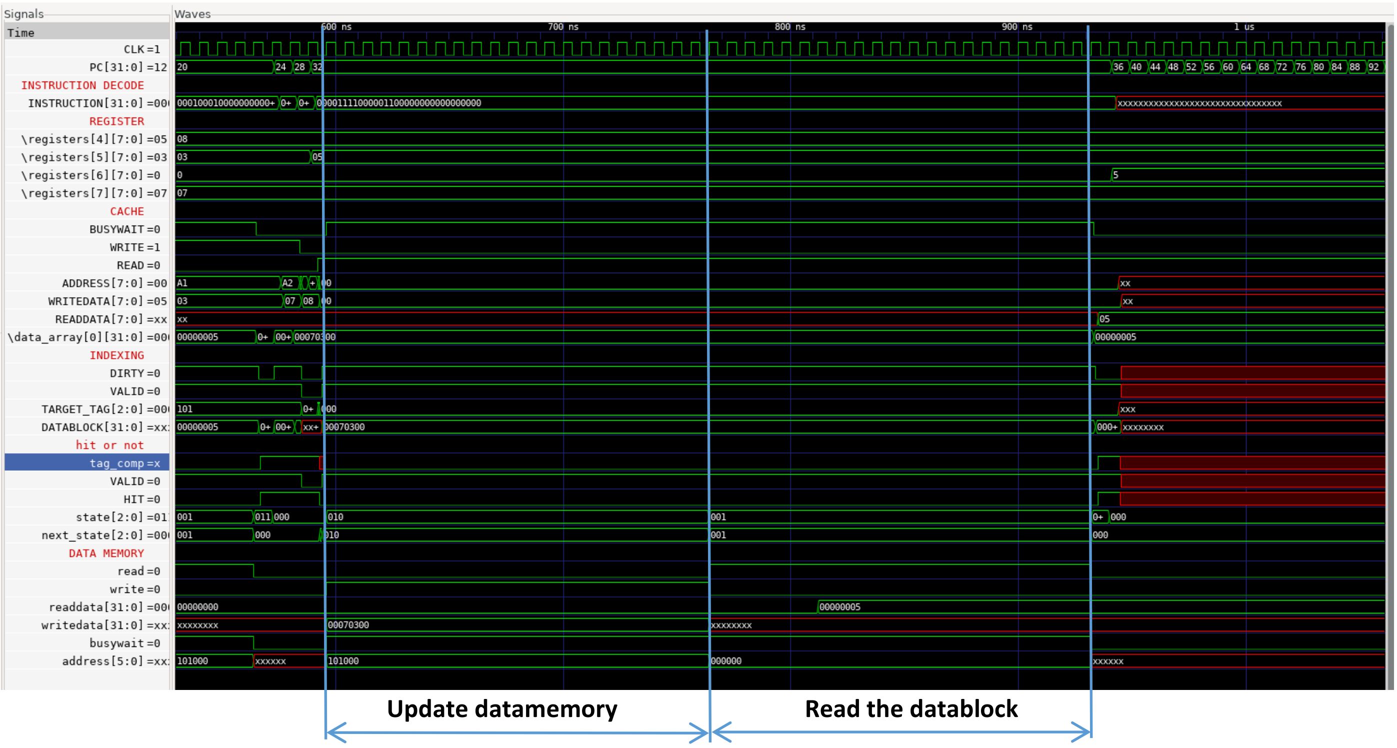
<u>Timing diagram - WITHOUT CACHE</u>



- lwi 6 0x00 directly accesses memory
- Loads one word (no cache, no write-back)

Total Delay = 5 cycles

Timing diagram - WITH CACHE



At this point:

- The cache currently holds the block for 0xA0–0xA3 (after **`swi 5 0xA1`** and **`swi 7 0xA2`**)
- That block is dirty, because we wrote to 0xA1 and 0xA2
- Now we're trying to load from 0x00, which maps to a different block
- Since it's a read miss and the current block is dirty, the cache must:
    - Write back the dirty block (0xA0–0xA3) to main memory.
    - Load the new block (0x00–0x03) from memory
    - Read the required word (0x00) from the cache

Timing (Assumptions):

- One word memory access = 5 cycles
- One block = 4 words

So:

- Write back dirty block = 4 words = 4 × 5 = 20 cycles
- Load new block = 4 words = 4 × 5 = 20 cycles

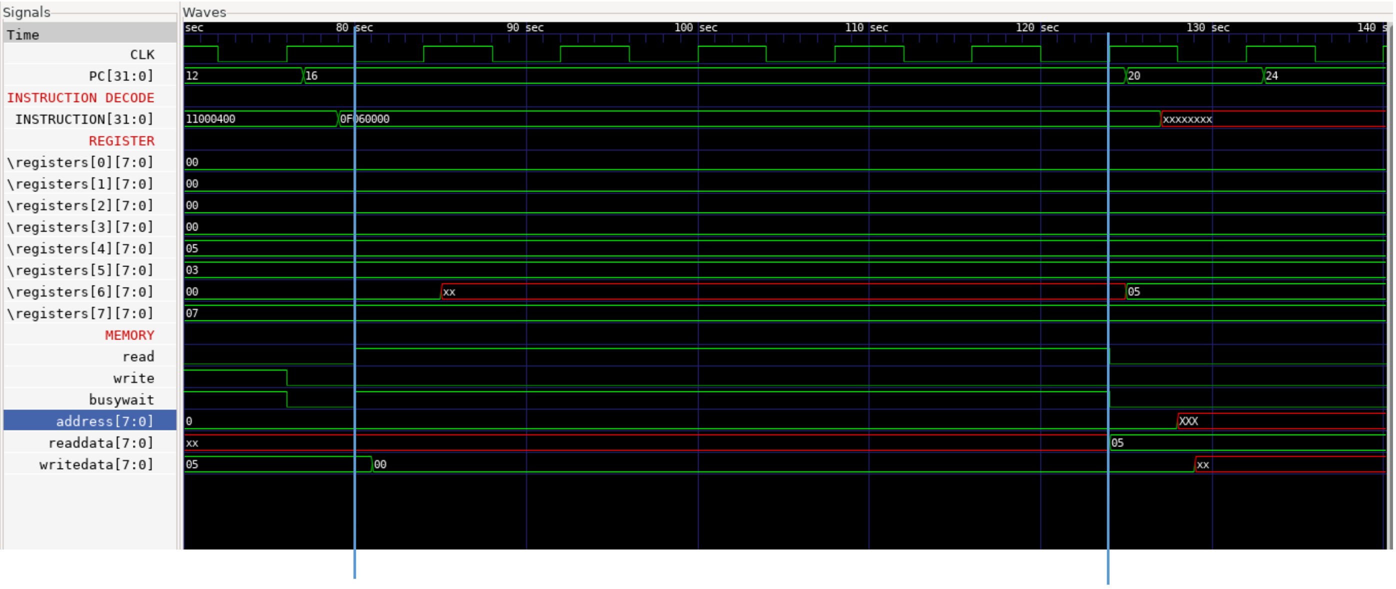
Total with cache = 20 + 20 = 40 cycles.

Case 6:Cache hit when read =1 and write = 0 (read hit)

Code

```
loadi 4 0x05
loadi 5 0x03
loadi 7 0x07
swi 4 0x00
lwi 6 0x00    //Read hit
```

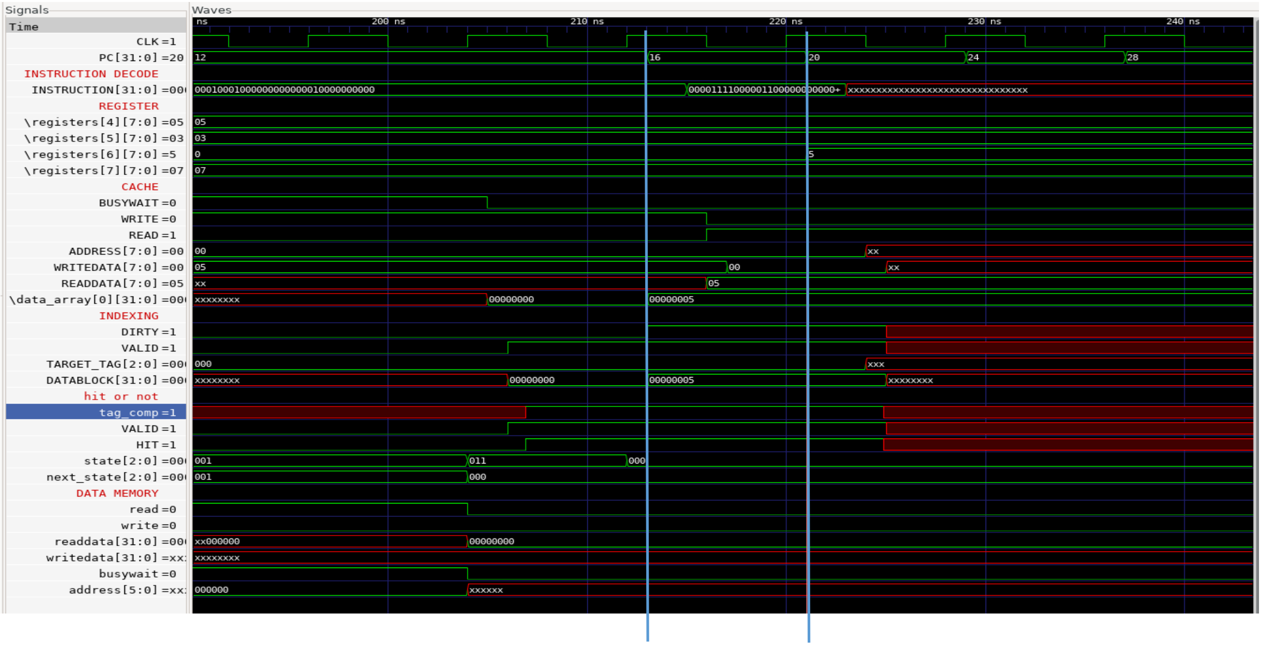
Timing diagram - WITHOUT CACHE



- If there's no cache, then **`lwi 6 0x00`** must access main memory directly.
- A memory read operation for a single word takes 20 cycles.

Total without cache = 20 cycles

Timing diagram - WITH CACHE



The instruction **swi 4 0x00** writes to memory address 0x00.

- Since it's the first access, the cache loads the block containing address 0x00 (e.g., block 0x00–0x03), sets valid = 1, and marks it dirty = 1 after the write.

The next instruction, **lwi 6 0x00**, reads from the same address.

- That address is already in the cache, so this is a read hit.
- The data is fetched immediately from the cache without accessing main memory.

Read hit = 1 cycle latency

No memory access or block transfer needed.

Final Conclusion

In this comparison, we analyzed various memory access scenarios in both cache-enabled and non-cached systems, focusing on write hits, write misses, read hits, and read misses.

- In cache hit cases (both read and write), the data is already present in the cache. These operations are completed within a single clock cycle, making with-cache access significantly faster than accessing main memory directly, which typically takes 5 cycles per word.

- However, in cache miss situations, especially when the cache needs to load an entire block (e.g., 4 words), the delay becomes substantial. For example:
    - A write miss or read miss with dirty = 0 takes around 20 cycles
    - A write miss or read miss with dirty = 1 requires writing back the old block and loading a new one, taking up to 40 cycles.

This shows that while cache improves performance for frequent accesses (hits), it introduces additional overhead during misses due to block-based memory transfers and write-back handling.