

GBANA DESIGN DOC

Version 0.1.0

April 25, 2025

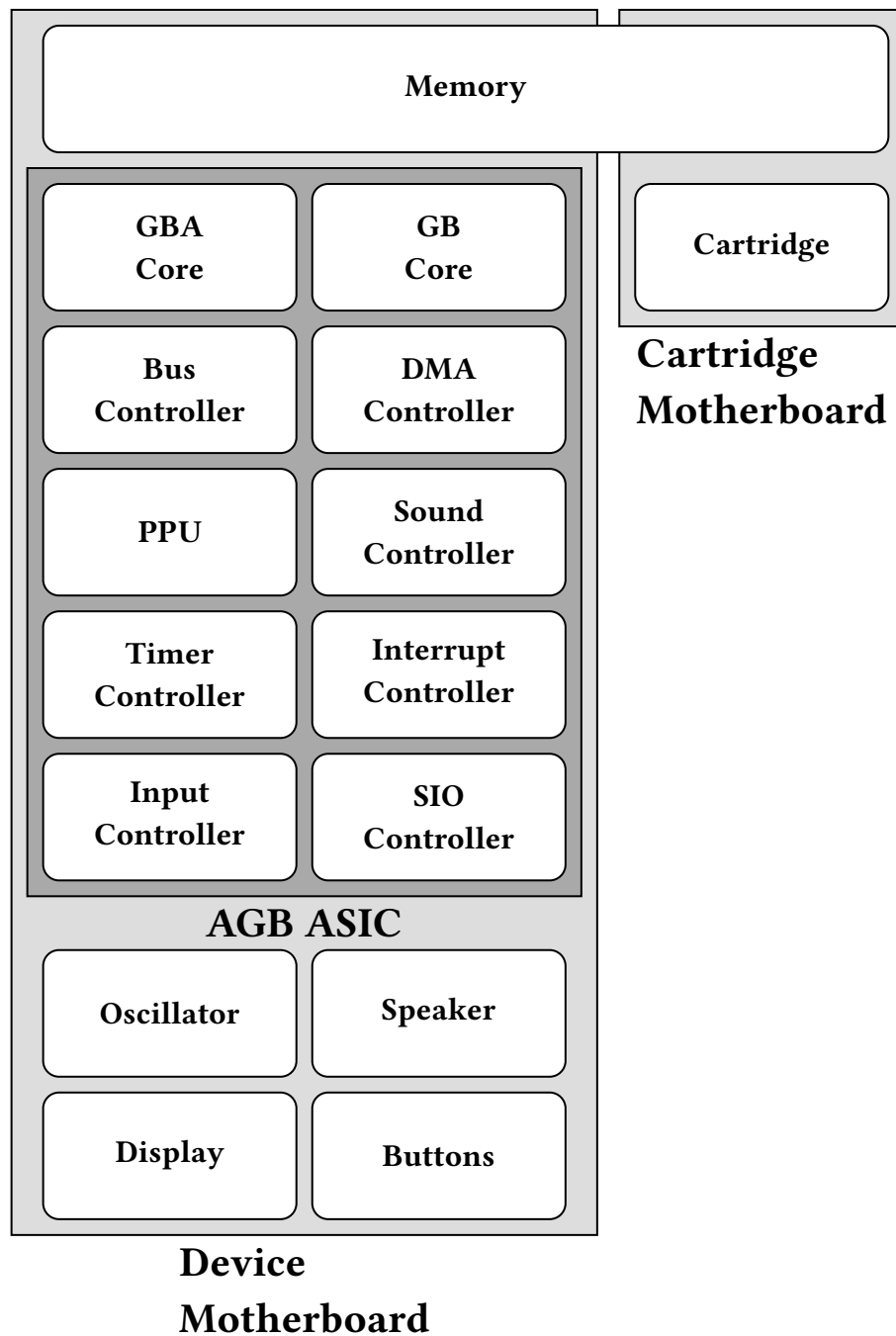
Contents

1. CLASSES	3
2. BLOCK DIAGRAM	4
3. EMULATING THE CLOCK & CYCLE	5
4. SIGNALS	6
5. SEQUENCES	8
5.1. MEMORY SEQUENCE	8
5.2. NONSEQUENTIAL MEMORY SEQUENCE	9
5.3. SEQUENTIAL MEMORY SEQUENCE	10
5.4. INTERNAL SEQUENCE	11
5.5. MERGED INTERNAL-SEQUENTIAL SEQUENCE	12
5.6. PIPELINED ADDRESSING	13
5.7. DEPIPELINED ADDRESSING	13
5.8. DATA WRITE SEQUENCE	13
5.9. DATA READ SEQUENCE	14
5.10. HALFWORD-WIDE MEMORY SEQUENCE	15
5.11. BYTE-WIDE MEMORY SEQUENCE	16
5.12. RESET SEQUENCE	18
5.13. GENERAL TIMING	19
5.14. ADDRESS BUS CONTROL	20
5.15. DATA BUS CONTROL	21
5.16. EXCEPTION CONTROL	21
5.17. ADDRESS LATCH CONTROL	22
5.18. ADDRESS PIPELINE CONTROL	22
5.19. GENERAL INSTRUCTION CYCLE	23
5.20. BRANCH AND BRANCH WITH LINK INSTRUCTION CYCLE	23
5.21. THUMB BRANCH WITH LINK INSTRUCTION CYCLE	24
5.22. BRANCH AND EXCHANGE INSTRUCTION CYCLE	25
5.23. DATA PROCESSING INSTRUCTION CYCLE	26
5.24. MULTIPLY AND MULTIPLY ACCUMULATE INSTRUCTION CYCLE	29
5.25. LOAD REGISTER INSTRUCTION CYCLE	33
5.26. STORE REGISTER INSTRUCTION CYCLE	35
5.27. LOAD MULTIPLE REGISTER INSTRUCTION CYCLE	36
5.28. STORE MULTIPLE REGISTER INSTRUCTION CYCLE	40
5.29. DATA SWAP INSTRUCTION CYCLE	42
5.30. SOFTWARE INTERRUPT AND EXCEPTION INSTRUCTION CYCLE	43
5.31. UNDEFINED INSTRUCTION CYCLE	45
5.32. UNEXECUTED INSTRUCTION CYCLE	46
Index of Figures	47

1. CLASSES

NAME	FILE	TYPE	DESCRIPTION
GBA Core	gba_core.odin	Singleton Class	The ARM7TDMI core inside the AGB ASIC.
GB Core	gb_core.odin	Singleton Class	The SM83 core inside the AGB ASIC.
Bus Controller	bus_controller.odin	Singleton Class	The bus control circuits inside the AGB ASIC.
DMA Controller	dma_controller.odin	Singleton Class	The direct memory access control circuits inside the AGB ASIC.
Sound Controller	sound_controller.odin	Singleton Class	The sound control circuits inside the AGB ASIC.
Memory	memory.odin	Singleton Class	The internal/device and the external/cartridge memory.
Buttons	buttons.odin	Singleton Class	The buttons and adjacent circuits on the device motherboard.
Display	display.odin	Singleton Class	The LCD display and adjacent circuits.
GBA Isa	gba_isa.odin	Helper Class	Implementation of the ARM4T ISA of the ARM7TDMI core.
GB Isa	gb_isa.odin	Helper Class	Implementation of the ISA of the Sharp SM83 core (a hybrid between the 8085 ISA and the Z80 ISA).
PPU	ppu.odin	Singleton Class	The Picture Processing Unit / LCD Video Controller inside the AGB ASIC.
Signal	line_and_bus.odin	Instance Class	A support class to emulate signals.
SIO Controller	sio_controller.odin	Singleton Class	The Serial Input/Output control circuits located inside the AGB ASIC.
Speakers	speakers.odin	Singleton Class	The audio output device and adjacent circuits on the device motherboard.
Util	util.odin	Helper Class	General utilities.

2. BLOCK DIAGRAM



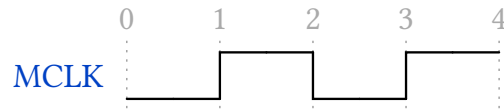
Each rounded rectangle is a **component**. Each component has a state object, which holds its data. Each non-rounded rectangle is a group of components that are semantically related. Components operate concurrently and share data only by means of **signals**.

3. EMULATING THE CLOCK & CYCLE

Related procedures:

`test_main_clock` Test procedure.

GBANA is phase-accurate. Every phase of every cycle is simulated, and the logic is verified phase-to-phase.



Main clock frequency: 16 MHz, ie approximately 62.5 ns per cycle. Each phase has two parts: a *start* part, where all the signals are updated and their callback functions are called, and an *interior* part, where the components execute their logic based on their internal state and the updated signals.

4. SIGNALS

Components may only communicate by means of **signals**. There are two ways to affect a signal: (1) by *putting* data on it, and (2) by *forcing* data on it. Forcing updates the output value immediately. Putting schedules an update to the output value, to occur after a certain number of ticks.

NAME	CLASS	COMPONENTS	DESCRIPTION
A	Memory Interface	Memory / Bus Controller	(Address Bus) The GBA Core / GB Core writes an address here when it requests memory access.
ABE	Bus Controls	Bus Controller	(Address Bus Enable) The GBA Core sets this to high to become master of the address bus (this is effectively a mutex on A), and to low to make the DMA Controller master of the address bus.
ABORT	Memory Management Interface	GBA Core	The Memory sets this to high when the requested memory operation cannot be performed, and to low when it can.
BIGEND	Bus Controls	GBA Core	The GBA Core sets this to high to interpret words in memory as being big-endian, and to low to interpret them as being little-endian. On the GBA, this is always 0 (little-endian format).
BL	Memory Interface	Memory / Bus Controller	(Byte Latch) The GBA Core writes a bit mask here to indicate which part of the requested word is to be read/written.
DBE	Bus Controls	Bus Controller	(Data Bus Enable) The GBA Core sets this to high to become master of the data output bus (this is effectively a mutex on DOUT), and to low to make DMA Controller master of the address bus.
DIN	Memory Interface	GBA Core / Bus Controller	(Unidirectional Data Input Bus) The Memory writes data here when a read request has been made.
DOUT	Memory Interface	Memory / Bus Controller	(Unidirectional Data Output Bus) The GBA Core writes data here when a write request has been made.
FIQ	Interrupts	GBA Core	(Fast Interrupt Request) Set this to high to request a fast interrupt.
ISYNC	Interrupts	GBA Core	(Synchronous Interrupt) Set this to high to indicate that the requested interrupt should be synchronous to the processor clock.
IRQ	Interrupts	GBA Core	(Interrupt Request) Set this to high to request an interrupt.

NAME	CLASS	COMPONENTS	DESCRIPTION
LOCK	Memory Interface	Memory / Bus Controller	The GBA Core / GB Core sets this to high to gain exclusive access to the Memory (this is effectively a mutex on the Memory signals).
MAS	Memory Interface	Memory / Bus Controller	(Memory Access Size) The GBA Core / GB Core writes here the size of the requested data access.
MCLK	Clocks and Timing	GBA Core, Memory	(Main Clock) The main clock.
M	Processor Mode	GBA Core	(Processor Mode) The current mode of the ARM7TDMI core.
MREQ	Memory Interface	Memory / Bus Controller	(Memory Request) The GBA Core / GB Core sets this to high to request memory access in the subsequent cycle.
OPC	Memory Interface	Memory	(Op-Code Fetch) The GBA Core / GB Core sets this to high to indicate that the requested memory access is to fetch the next instruction.
RESET	Bus Controls	GBA Core	Set this to high to initialize / restart the ARM7TDMI processor.
RW	Memory Interface	Memory / Bus Controller	(Read/Write) The GBA Core / GB Core sets this to high to indicate that the requested memory access is a read, and to low to indicate that it is a write.
SEQ	Memory Interface	Memory / Bus Controller	(Sequential Cycle) The GBA Core / GB Core sets this to high to indicate that the subsequent memory cycle will be sequential, and to low to indicate that it will be nonsequential.
TBIT	Processor State	GBA Core	(Thumb Mode Bit) Set this to high to switch the ARM7TDMI core to Thumb mode, and to low to switch it to ARM mode.
WAIT	Clocks and Timing	GBA Core	Set this to high to insert a wait cycle.

5. SEQUENCES

There are two fundamental types of sequence: **internal sequence** and **external sequence**. Internal cycles are initiated by the **GBA Core** by calling a `gba_initiate_<cycle_name>_cycle_request` procedure and then one or more other components respond by interpreting the signals and calling a `<component_name>_initiate_<cycle_name>_cycle_response` procedure.

Types of intervals in a timing diagram:

- *Open Unshaded* - The line/bus is expected to remain stable throughout this interval.
 - *Writing* occurs at the *start* and is prohibited in the *interior*.
 - *Reading* is allowed at the *start* (by signals succeeding it in the tick order) and in the *interior*.
- *Open Shaded* - The line/bus is expected to change at an arbitrary time during this interval.
 - *Writing* is prohibited at the *start* and allowed in the *interior*.
 - *Reading* is allowed at the *start* and prohibited in the *interior*.
- *Closed* - The line/bus is disabled.
 - *Writing* is prohibited at the *start* and prohibited in the *interior*.
 - *Reading* is prohibited at the *start* and in the *interior*.

In request/response contexts, request data is displayed in blue, and response data is displayed in pink.

5.1. MEMORY SEQUENCE

RELATED PROCEDURES	
<code>test_memory_sequence</code>	Test procedure.
<code>gba_request_memory_sequence</code>	Request procedure, must be called during phase 1.
<code>memory_respond_memory_sequence</code>	Response procedure, must be called 1 cycle after the request procedure.

REQUEST SIGNALS	RESPONSE SIGNALS
<code>MREQ</code> , <code>SEQ</code> , <code>RW</code> , <code>A</code> , <code>DOUT</code>	<code>WAIT</code> , <code>ABORT</code> , <code>DIN</code>

The Memory Sequence is an external sequence where the GBA Core requests and the Memory responds. Reads and writes have distinct logic. Word-wide bus access, halfword-wide bus access, and byte-wide bus access have distinct logic. The Memory may extend the time to fulfill the request and it may assert that a request may not be fulfilled.

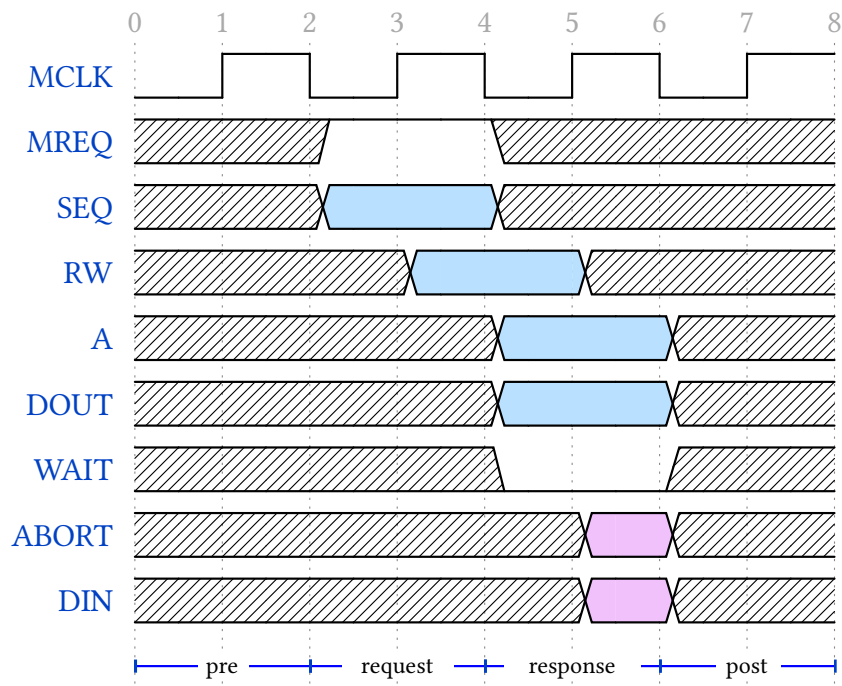


Figure 1: Memory Sequence timing diagram

- The **GBA Core** must set **SEQ** to high when the access is sequential to the access performed in the previous cycle.
- The **GBA Core** must set **RW** to high for reading and to low for writing.
- The **GBA Core** must write the address to **A**.
- The **Memory** may set **WAIT** to high to delay the response cycle.
- The **Memory** may set **ABORT** to high to indicate that the request cannot be fulfilled.
- The **Memory** must put the data on **DIN** during the high phase of the response cycle.

*Insert Delayed Memory Sequence, same as Memory Sequence but with **WAIT** set to high for a few cycles.*

5.2. NONSEQUENTIAL MEMORY SEQUENCE

RELATED PROCEDURES	
test_n_cycle	Test procedure.
gba_request_n_cycle	Request procedure, must be called during phase 1.
memory_respond_n_cycle	Response procedure, must be called 1 cycle after the request procedure.

REQUEST SIGNALS	RESPONSE SIGNALS
MREQ , SEQ , RW , A , DOUT	WAIT , ABORT , DIN

The Nonsequential Memory Sequence (or N-Cycle) is a memory access sequence, preceded by an internal sequence or another memory access sequence to an address other than the address immediately before the current address.

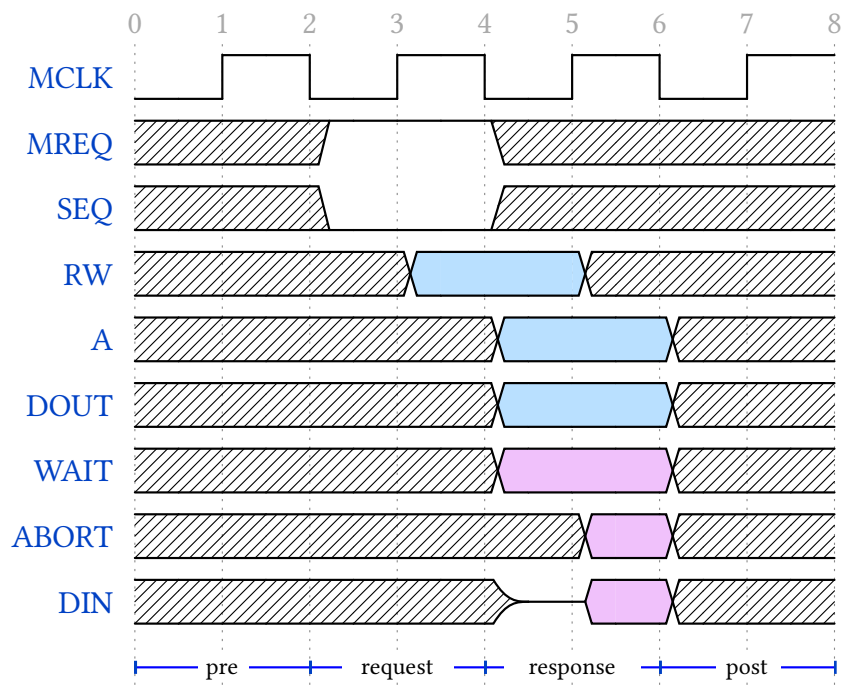


Figure 2: Nonsequential Memory Sequence timing diagram

- The **GBA Core** must set **RW** to high for reading and to low for writing.
- The **GBA Core** must write the address to **A**.
- The **Memory** may set **WAIT** to high to delay the response cycle.
- The **Memory** may set **ABORT** to high to indicate that the request cannot be fulfilled.
- The **Memory** must put the data on **DIN** during the high phase of the response cycle.

5.3. SEQUENTIAL MEMORY SEQUENCE

RELATED PROCEDURES	
test_s_cycle	Test procedure.
gba_request_s_cycle	Request procedure, must be called during phase 1.
memory_respond_s_cycle	Response procedure, must be called 1 cycle after the request procedure.

REQUEST SIGNALS	RESPONSE SIGNALS
MREQ, SEQ, RW, A, DOUT	WAIT, ABORT, DIN

The Sequential Memory Sequence (or S-Cycle) is a memory access sequence, preceded by another memory access sequence to the address immediately before the current address.

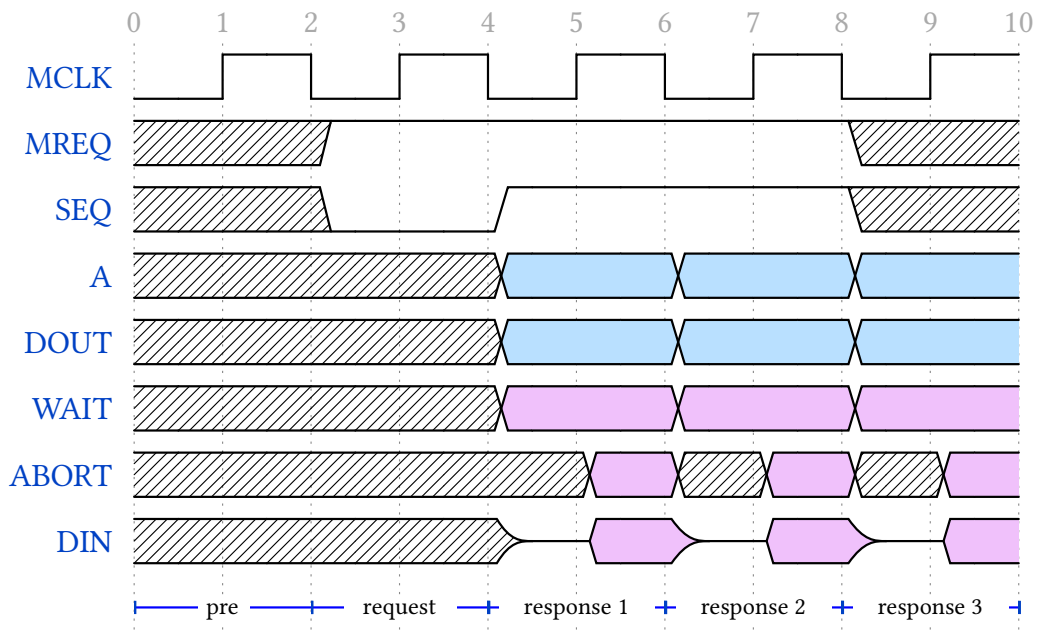


Figure 3: Sequential Memory Sequence timing diagram

- The **GBA Core** must set **RW** to high for reading and to low for writing.
- The **GBA Core** must write the address to **A**.
- The **Memory** may set **WAIT** to high to delay the next response cycle.
- The **Memory** may set **ABORT** to high to indicate that the request cannot be fulfilled.
- The **Memory** must put the data on **DIN** during the high phase of the response cycle.

5.4. INTERNAL SEQUENCE

RELATED PROCEDURES	
test_i_cycle	Test procedure.
gba_initiate_i_cycle	Initiation procedure, must be called during phase 1.

SIGNALS	
MREQ , SEQ , A , DIN , DOUT	

The Internal Sequence (or I-Cycle) is a sequence that doesn't involve exchanging data with any components outside of the core.

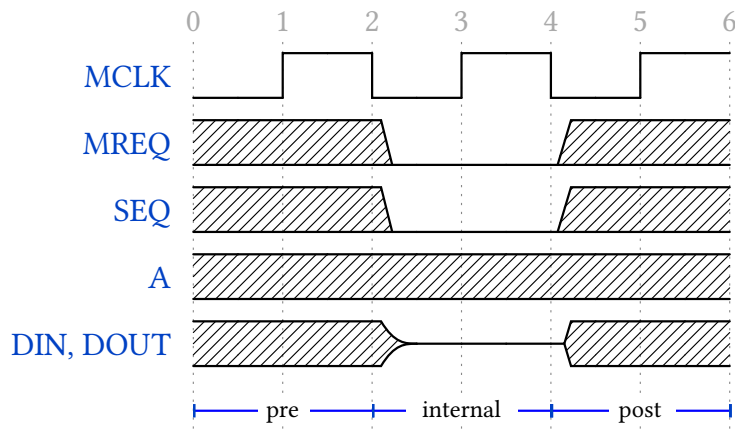


Figure 4: Internal Sequence timing diagram

5.5. MERGED INTERNAL-SEQUENTIAL SEQUENCE

RELATED PROCEDURES	
test_merged_is_cycle	Test procedure.
gba_request_merged_is_cycle	Request procedure, must be called during phase 1 of the request cycle.
memory_respond_merged_is_cycle	Response procedure, must be called during phase 1 of the response cycle.

REQUEST SIGNALS	RESPONSE SIGNALS
MREQ , SEQ , RW , A , DOUT	WAIT , ABORT , DIN

The Merged Internal-Sequential Sequence (or Merged IS-Cycle) is an internal sequence followed immediately by a sequential memory cycle.

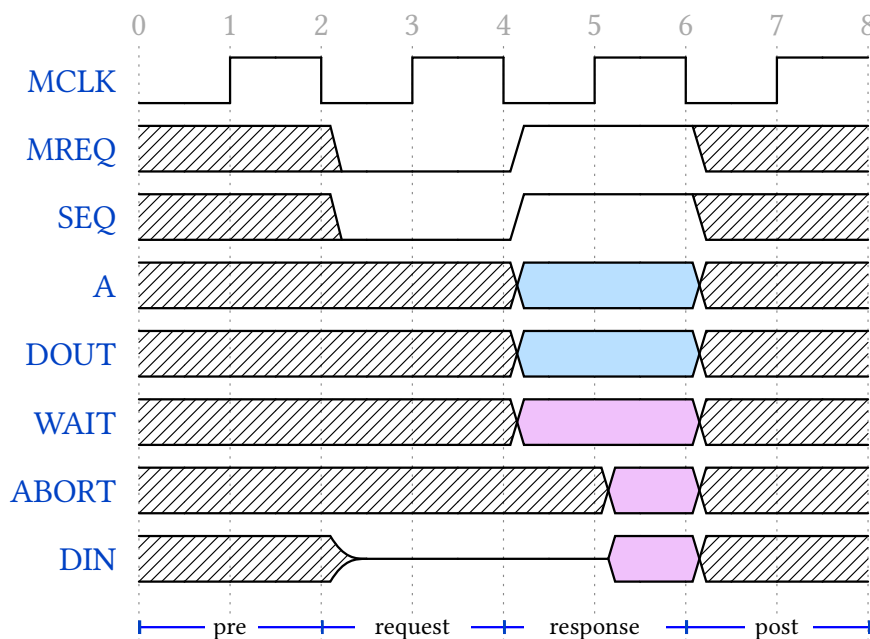


Figure 5: Merged Internal-Sequential Sequence timing diagram

5.6. PIPELINED ADDRESSING

This is irrelevant to the emulator, since the GBA uses SRAM and the intended address timing for SRAM is depipelined.

5.7. DEPIPELINED ADDRESSING

RELATED PROCEDURES	
test_depipelined_addressing	Test procedure.

REQUEST SIGNALS	RESPONSE SIGNALS
MREQ , SEQ , RW , A , DOUT	WAIT , ABORT , DIN

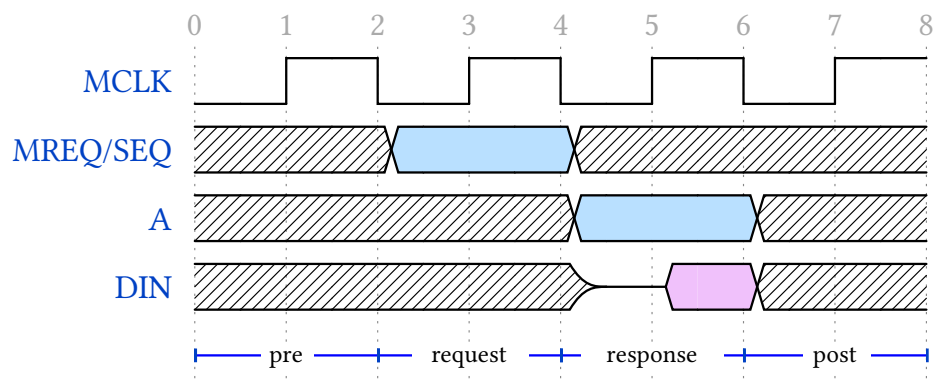


Figure 6: Depipelined Addressing timing diagram

5.8. DATA WRITE SEQUENCE

RELATED PROCEDURES	
test_data_write_sequence	Test procedure.
gba_request_data_write_cycle	Request procedure, must be called during phase 1 of the request cycle.
memory_respond_data_write_cycle	Response procedure, must be called during phase 1 of the response cycle.

REQUEST SIGNALS	RESPONSE SIGNALS
MREQ , RW , A , DOUT	WAIT , ABORT

A Data Write Sequence is an external sequence where a write operation is requested by the GBA Core.

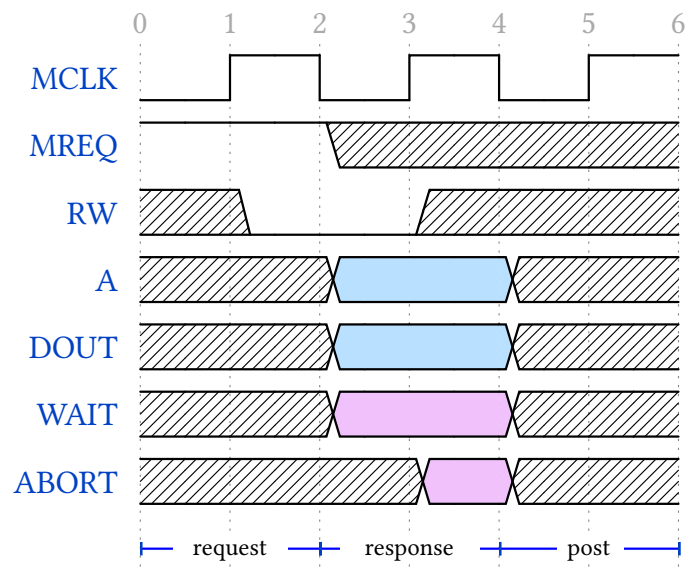


Figure 7: Data Write Sequence timing diagram

5.9. DATA READ SEQUENCE

RELATED PROCEDURES	
test_data_read_sequence	Test procedure.
gba_request_data_read_cycle	Request procedure, must be called during phase 1 of the request cycle.
memory_respond_data_read_cycle	Response procedure, must be called during phase 1 of the response cycle.

REQUEST SIGNALS	RESPONSE SIGNALS
MREQ , RW , A , DOUT	WAIT , ABORT

A Data Read Sequence is an external sequence where a read operation is requested by the GBA Core.

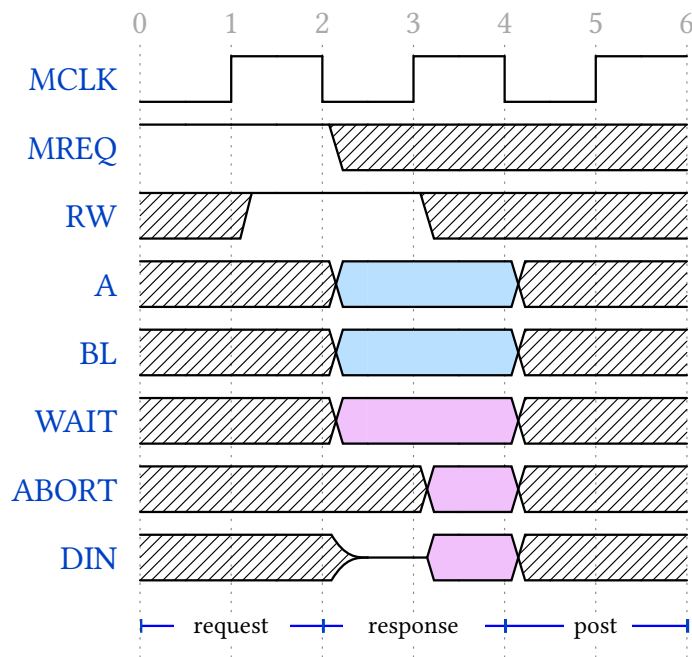


Figure 8: Data Read Sequence timing diagram

5.10. HALFWORD-WIDE MEMORY SEQUENCE

RELATED PROCEDURES	
test_halfword_memory_sequence	Test procedure.
gba_request_halfword_memory_sequence	Request procedure, must be called during phase 1.
memory_respond_halfword_memory_sequence	Response procedure, must be called 1 cycle after the request procedure.

REQUEST SIGNALS	RESPONSE SIGNALS
MREQ, SEQ, RW, A, DOUT	WAIT, ABORT, DIN

The Halfword-Wide Memory Sequence is the same as the basic Memory Sequence, except for memory with a 16-bit wide bus, thus requiring 2 cycles per word.

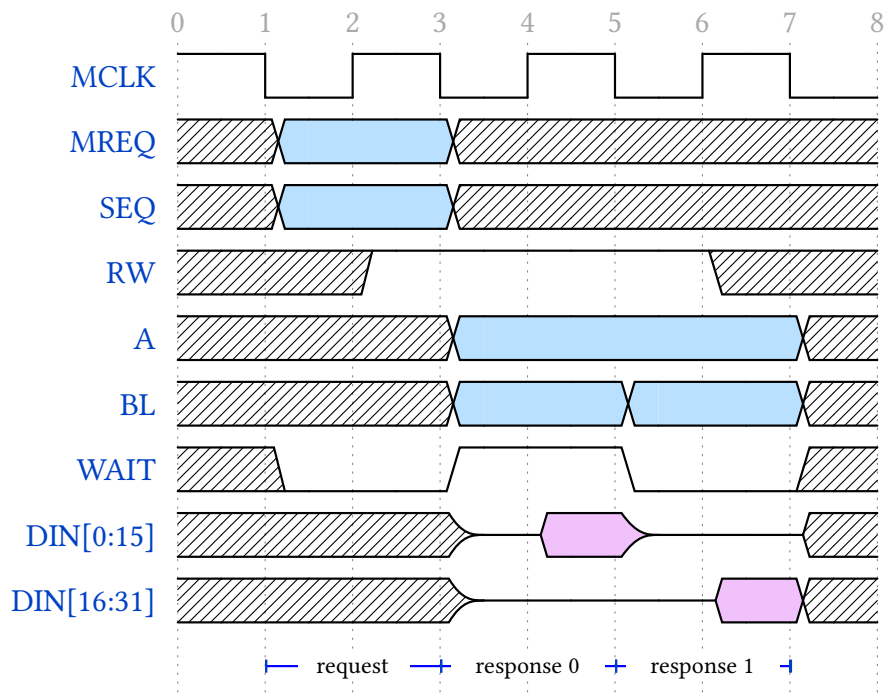


Figure 9: Halfword-Wide Read Memory Sequence timing diagram

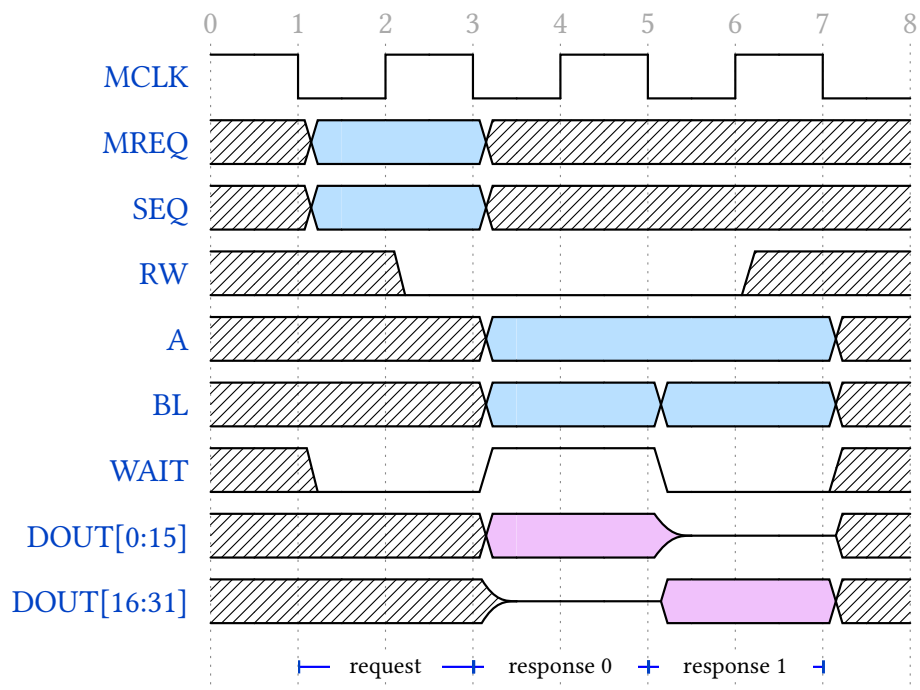


Figure 10: Halfword-Wide Write Memory Sequence timing diagram

5.11. BYTE-WIDE MEMORY SEQUENCE

RELATED PROCEDURES	
<code>test_byte_memory_sequence</code>	Test procedure.
<code>gba_request_byte_memory_sequence</code>	Request procedure, must be called during phase 1.

RELATED PROCEDURES	
memory_respond_byte_memory_sequence	Response procedure, must be called 1 cycle after the request procedure.

REQUEST SIGNALS	RESPONSE SIGNALS
MREQ , SEQ , RW , A , DOUT	WAIT , ABORT , DIN

The Byte-Wide Memory Sequence is the same as the basic Memory Sequence, except for memory with an 8-bit wide bus, thus requiring 4 cycles per word.

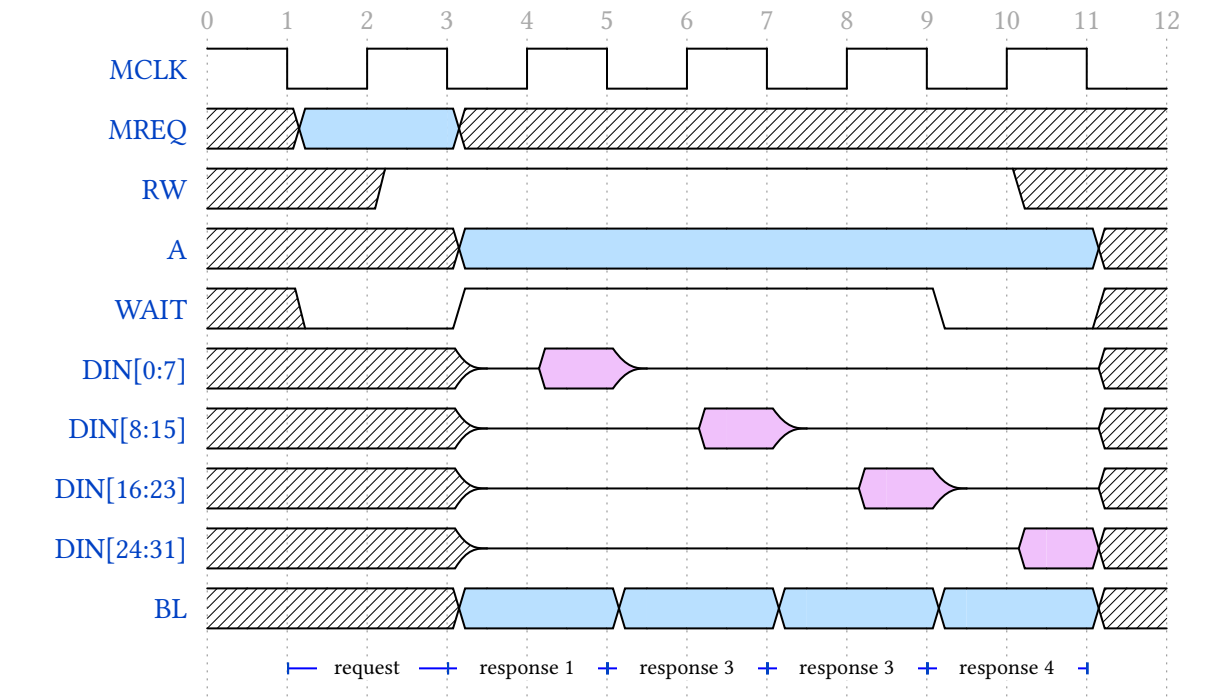


Figure 11: Byte-Wide Read Memory Sequence timing diagram

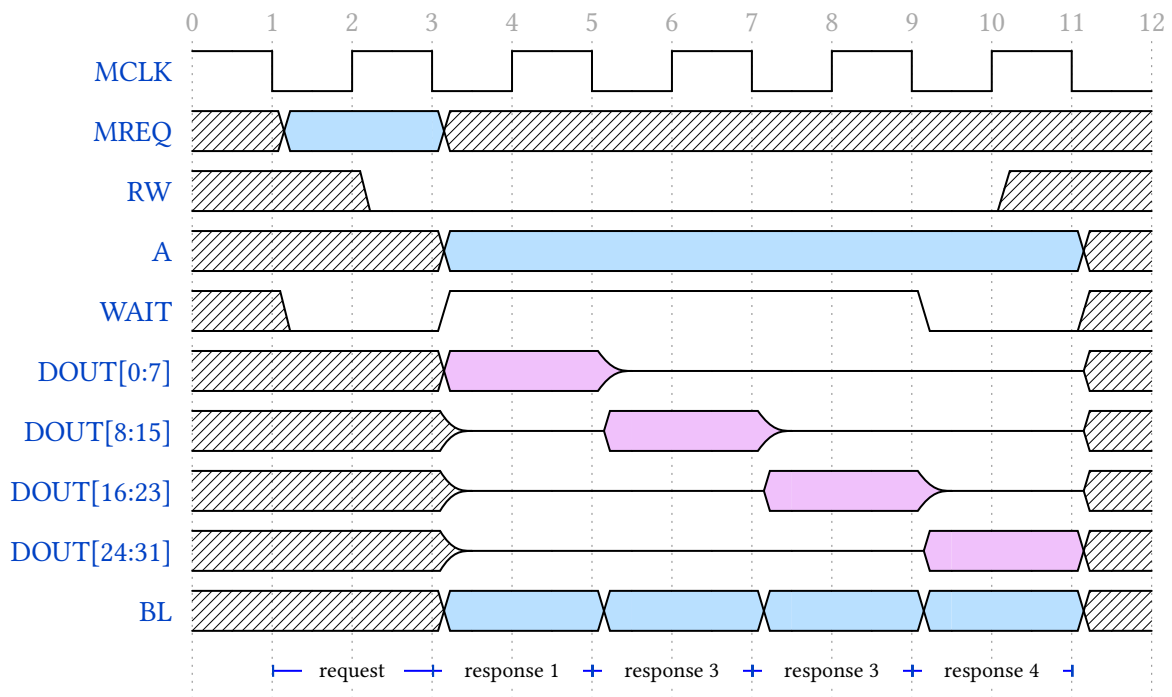


Figure 12: Byte-Wide Write Memory Sequence timing diagram

5.12. RESET SEQUENCE

RELATED PROCEDURES	
test_reset_sequence	Test procedure.
gba_initiate_reset_sequence	Initiation procedure, must be called during phase 1 of the very first cycle, and during phase of any cycle in which the reset button was pressed.

REQUEST SIGNALS	RESPONSE SIGNALS
MREQ , SEQ , RW , EXEC , OPC , A	WAIT , ABORT , DIN

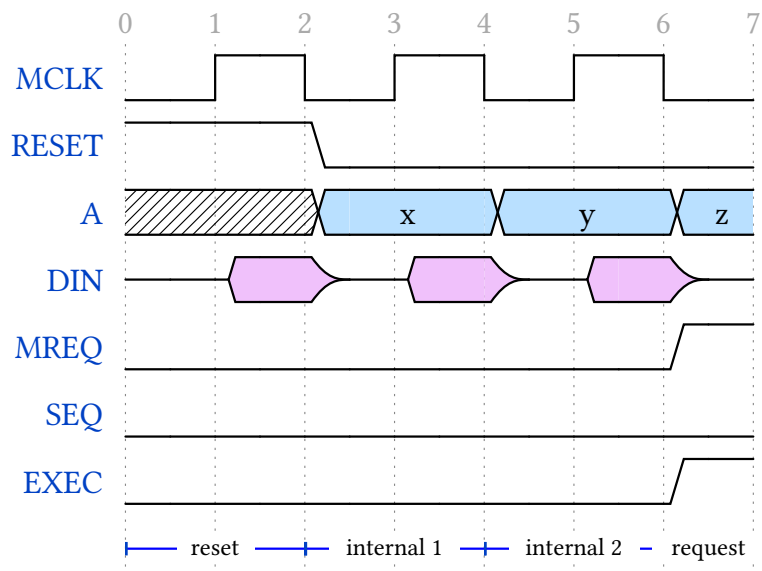


Figure 13: Reset Sequence timing diagram

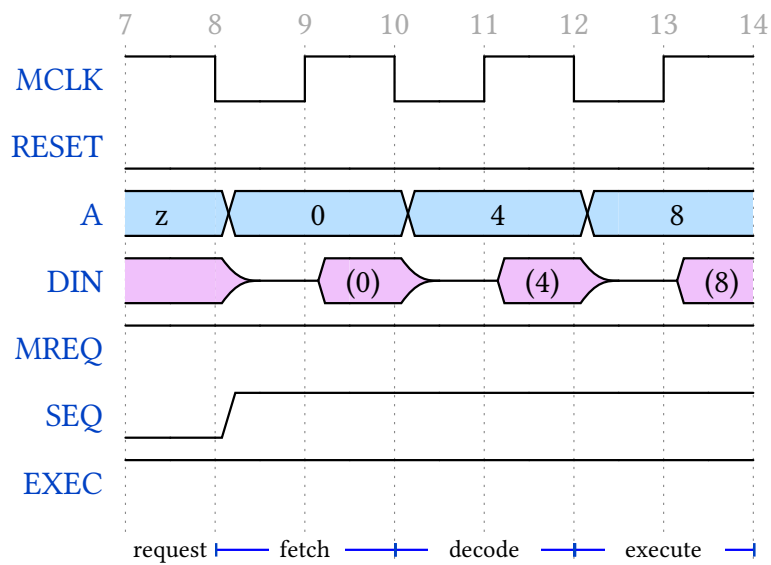


Figure 14: Reset Sequence timing diagram (continued)

5.13. GENERAL TIMING

RELATED PROCEDURES	
test_general_timing	Test procedure.
PHASE 1 SIGNALS	PHASE 2 SIGNALS
MREQ , SEQ , EXEC , EXEC , INSTRVALID , A , BIGEND , ISYNC	RW , MAS , LOCK , M , TBIT , OPC , ISYNC

The General Timing diagram defines in which phase of the cycle each of the control signals is allowed to change.

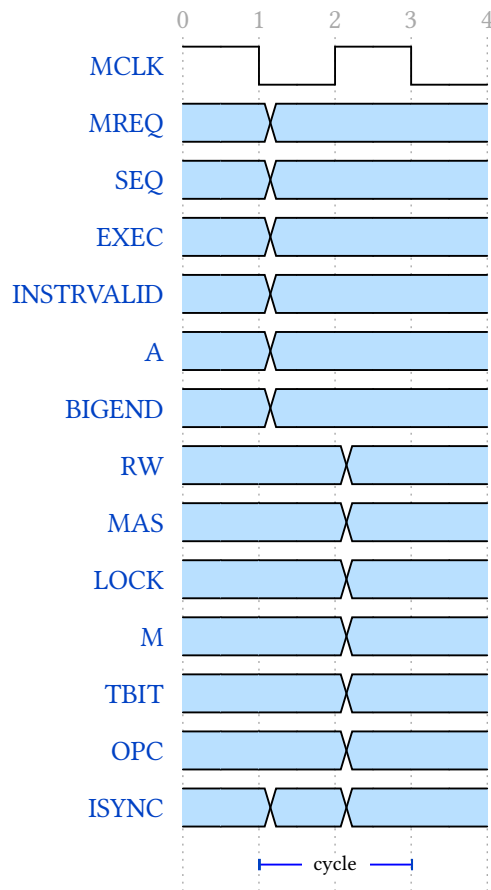


Figure 15: General timing diagram

5.14. ADDRESS BUS CONTROL

RELATED PROCEDURES	
test_address_bus_control	Test procedure.

SIGNALS	
ABE , A , RW , LOCK , OPC , MAS	

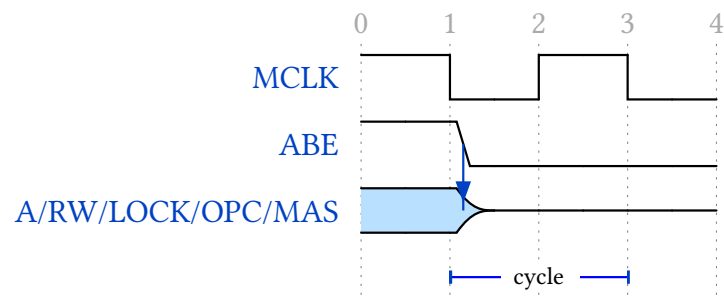


Figure 16: Address Bus Control timing diagram

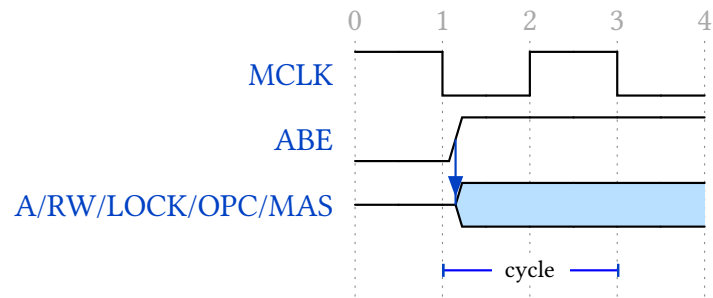


Figure 17: Address Bus Control timing diagram

- ABE may change during phase 1.
- A , RW , LOCK , OPC , and MAS are enabled/disabled immediately when ABE switches to high/low.
- A , RW , LOCK , OPC , and MAS must be stable at the starts of both phases.

5.15. DATA BUS CONTROL

RELATED PROCEDURES	
test_data_bus_control	Test procedure.

SIGNALS	
DBE , DIN , DOUT	

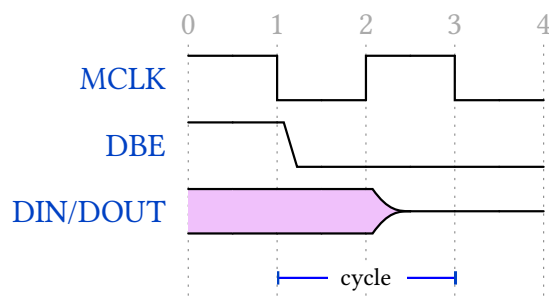


Figure 18: Data Bus Control timing diagram

5.16. EXCEPTION CONTROL

RELATED PROCEDURES	
test_exception_control	Test procedure.

SIGNALS	
ABORT , FIQ , IRQ , RESET	

The exceptional behaviors are: (1) aborted memory access, (2) interrupt, (3) fast interrupt, (4) reset.

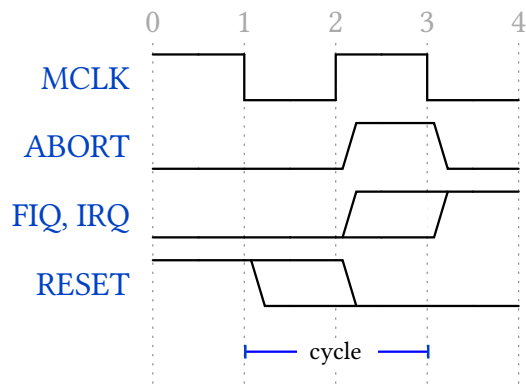


Figure 19: Exception Control timing diagram

1. FIQ and IRQ signals must be set one cycle ahead of the cycle in which they'll be handled, during the high phase, and they must remain stable through the start of the low phase of the next cycle.

5.17. ADDRESS LATCH CONTROL

This is irrelevant for the emulator.

5.18. ADDRESS PIPELINE CONTROL

RELATED PROCEDURES	
test_address_pipeline_control	Test procedure.

SIGNALS	
APE , A , RW , LOCK , OPC , MAS	

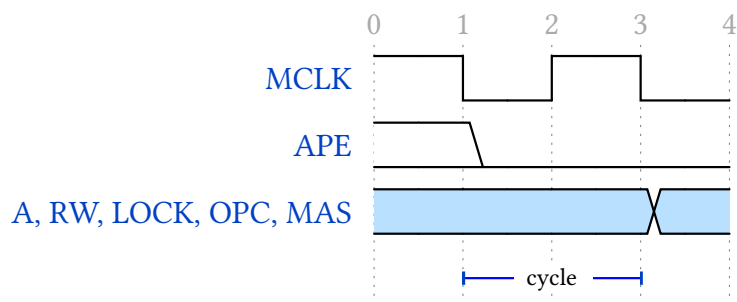


Figure 20: Address Pipeline Control timing diagram

5.19. GENERAL INSTRUCTION CYCLE

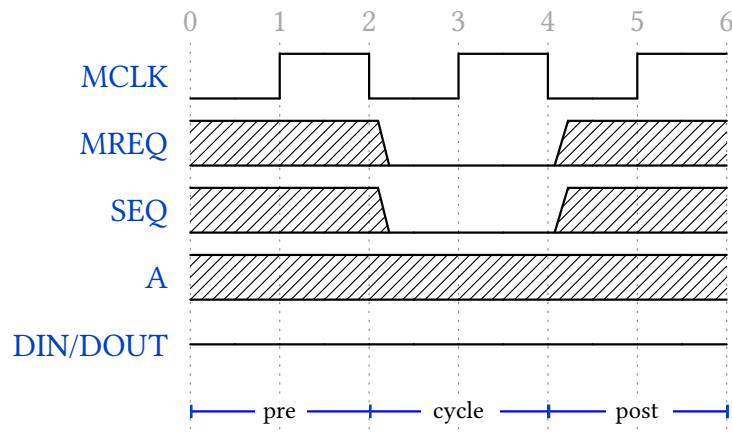


Figure 21: General Instruction Cycle timing diagram

1. There are two types of request signals: request type signals and request address signals, which are broadcast at least one tick ahead of the response cycle.
2. The request type signals (MREQ and SEQ) are pipelined up to 2 ticks ahead of the cycle to which they apply.
3. The request address signals (A , MAS , RW , OPC , and TBIT) are pipelined up to 1 tick ahead of the cycle to which they apply.
4. The instruction cycle is the response cycle.
5. When OPC is high, the address is incremented each cycle (epistemic status: *guess*).

5.20. BRANCH AND BRANCH WITH LINK INSTRUCTION CYCLE

RELATED INSTRUCTIONS

B , BL

RELATED PROCEDURES

test_branch_and_branch_with_link_instruction_cycle	Test procedure.
gba_request_branch_and_branch_with_link_instruction_cycle	Request procedure, must be called during phase 1.
memory_respond_branch_and_branch_with_link_instruction_cycle	Response procedure, must be called during phase 1.

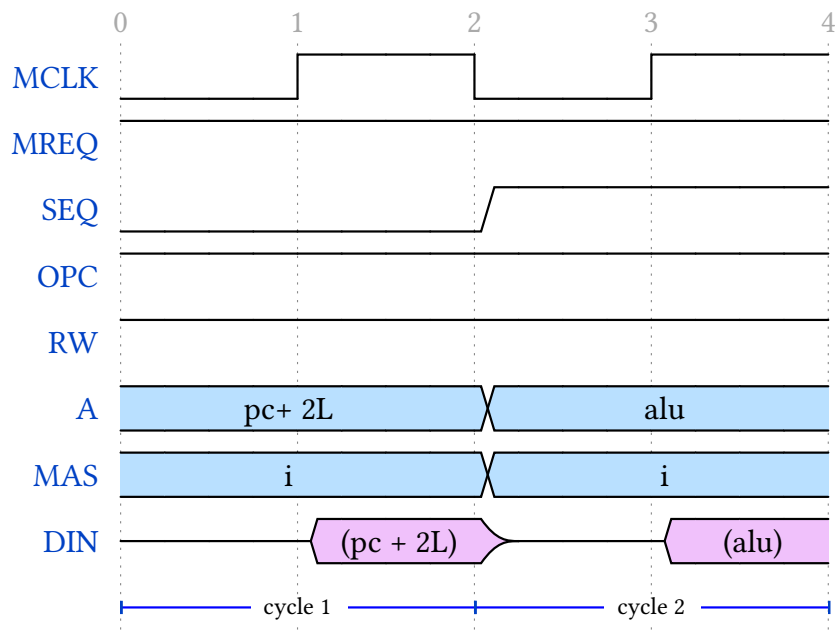


Figure 22: Branch and Branch with Link Instruction Cycle

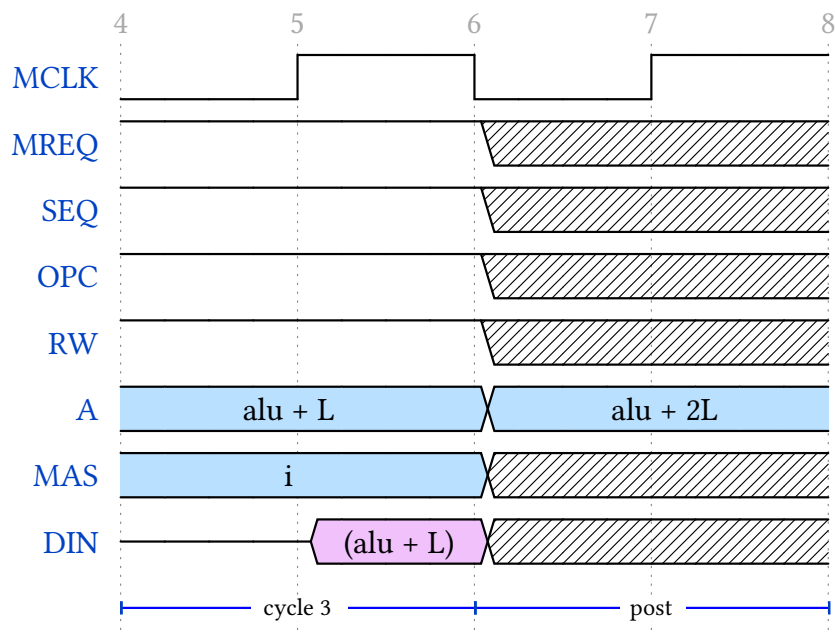


Figure 23: Branch and Branch with Link Instruction Cycle (continued)

5.21. THUMB BRANCH WITH LINK INSTRUCTION CYCLE

RELATED INSTRUCTIONS

RELATED PROCEDURES	
test_thumb_branch_with_link_instruction_cycle	Test procedure.
gba_request_thumb_branch_with_link_instruction_cycle	Request procedure.
memory_respond_thumb_branch_with_link_instruction_cycle	Response procedure.

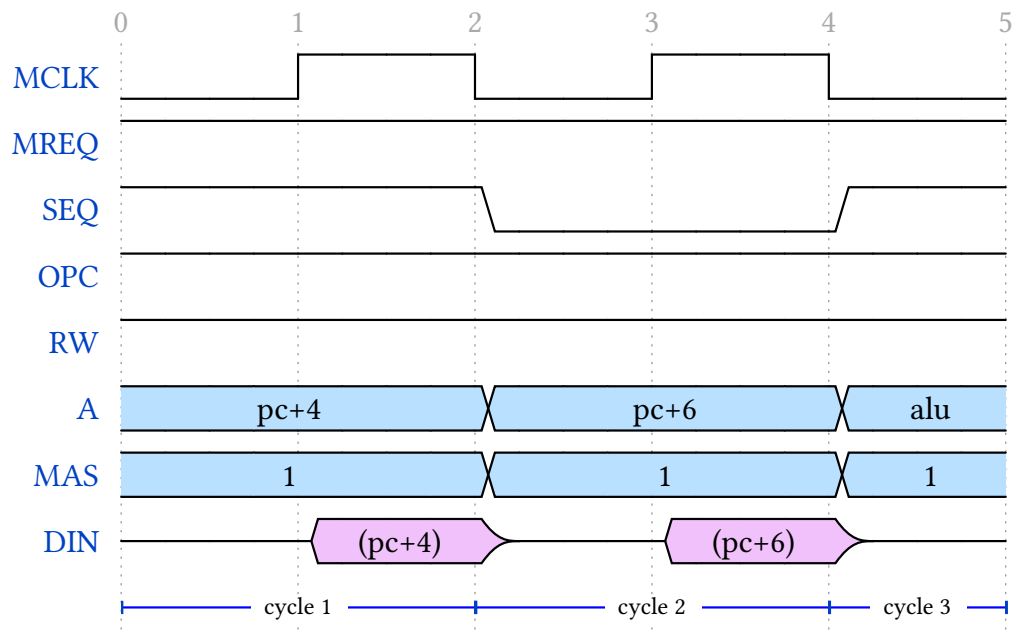


Figure 24: Thumb Branch with Link Instruction Cycle

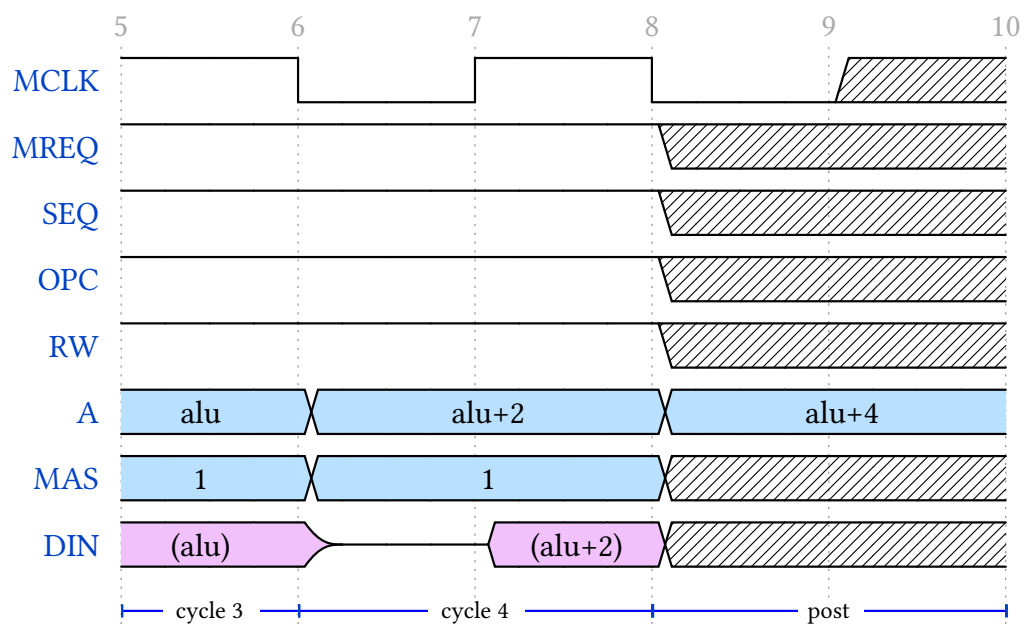


Figure 25: Thumb Branch with Link Instruction Cycle (continued)

5.22. BRANCH AND EXCHANGE INSTRUCTION CYCLE

RELATED INSTRUCTIONS	
BX	

RELATED PROCEDURES	
test_branch_and_exchange_instruction_cycle	Test procedure.
gba_request_branch_and_exchange_instruction_cycle	Request procedure.
memory_respond_branch_and_exchange_instruction_cycle	Response procedure.

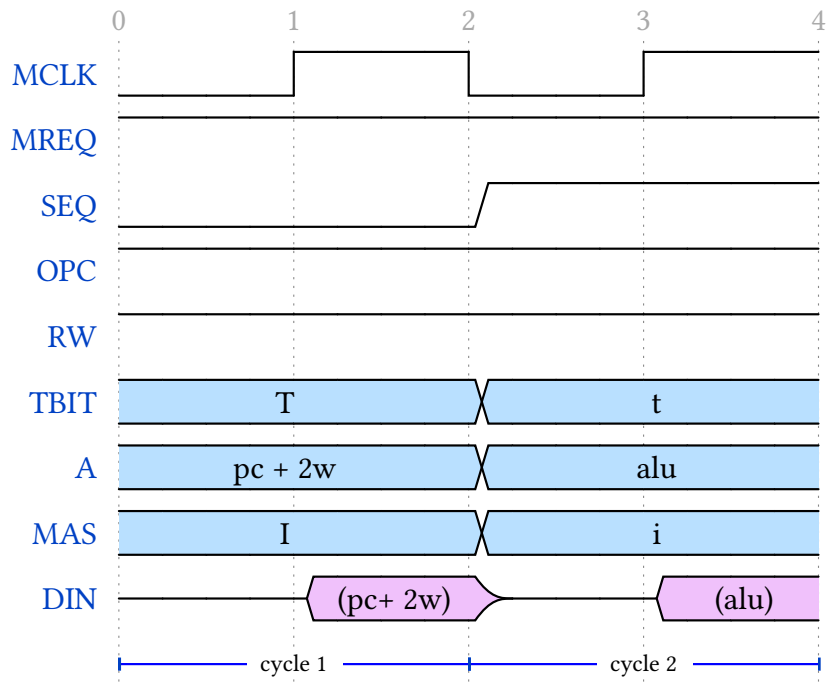


Figure 26: Branch and Exchange Instruction Cycle

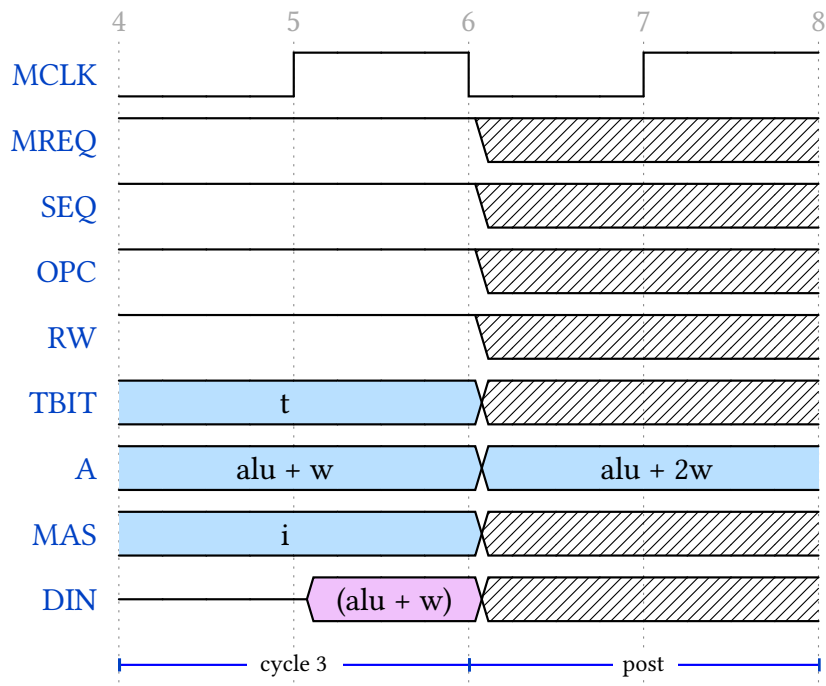


Figure 27: Branch and Exchange Instruction Cycle (continued)

5.23. DATA PROCESSING INSTRUCTION CYCLE

RELATED INSTRUCTIONS

ADC , ADD , AND , BIC , CMN , CMP , EOR , MOV , MRS , MSR , MVN , ORR , RSB , RSC , SBC , SUB , TEQ , TST

RELATED PROCEDURES	
test_data_processing_instruction_cycle	Test procedure.
gba_request_data_processing_instruction_cycle	Request procedure.
memory_respond_data_processing_instruction_cycle	Response procedure.

It seems like whenever SEQ is high, the address will always increment in the post cycle.

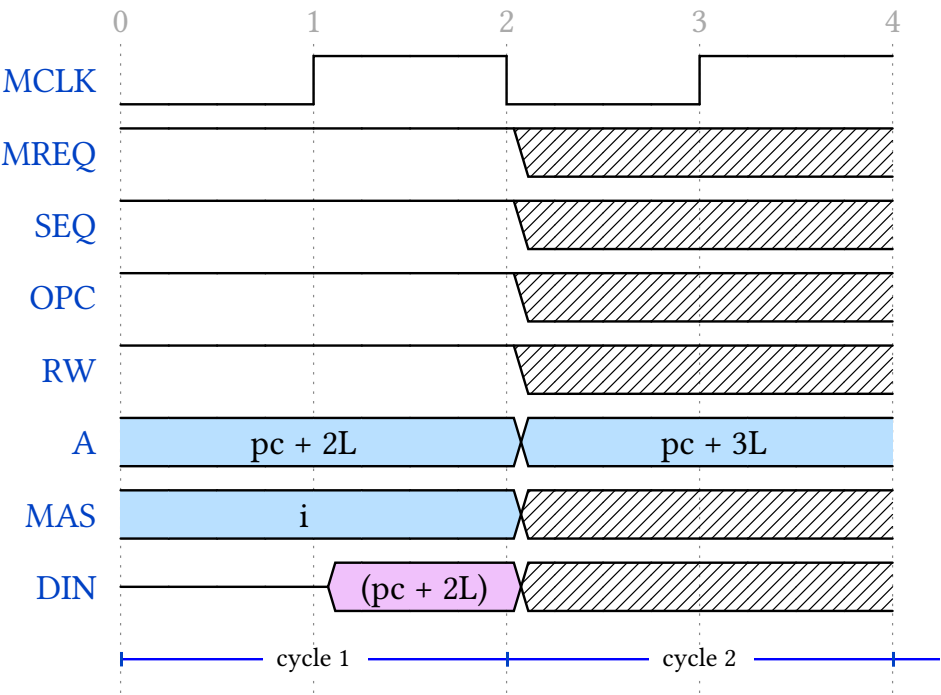


Figure 28: Data Processing Instruction Cycle (normal)

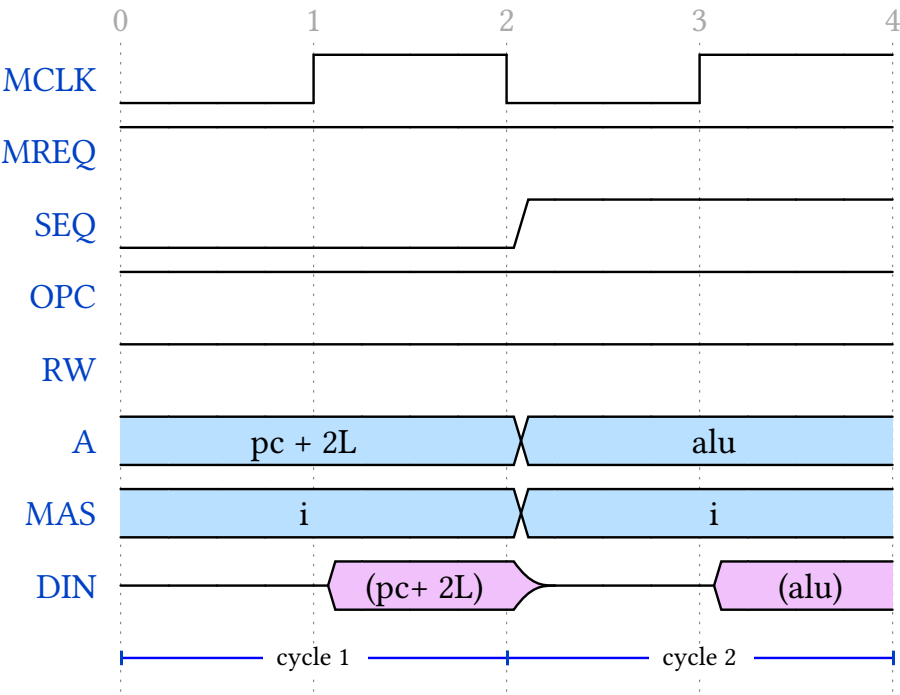


Figure 29: Data Processing Instruction Cycle (dest=pc)

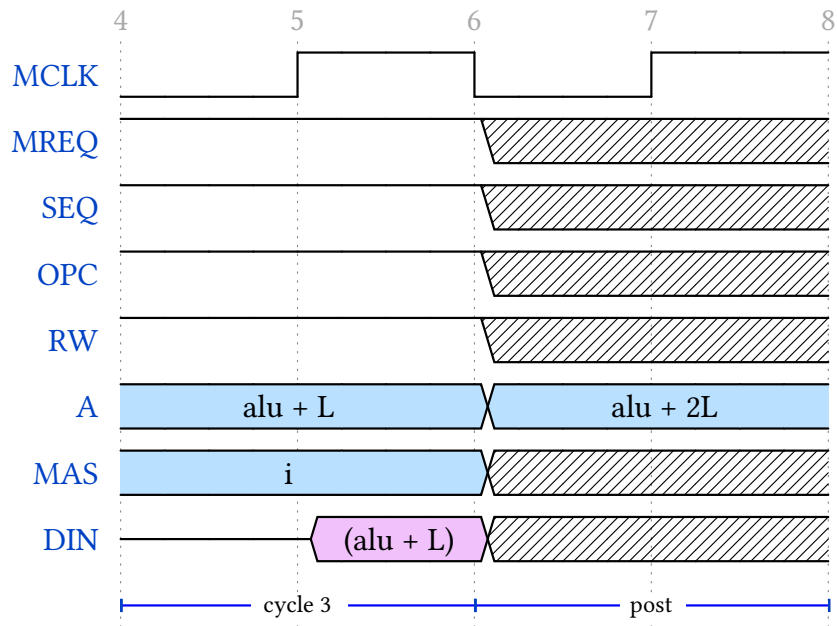


Figure 30: Data Processing Instruction Cycle (dest=pc) (continued)

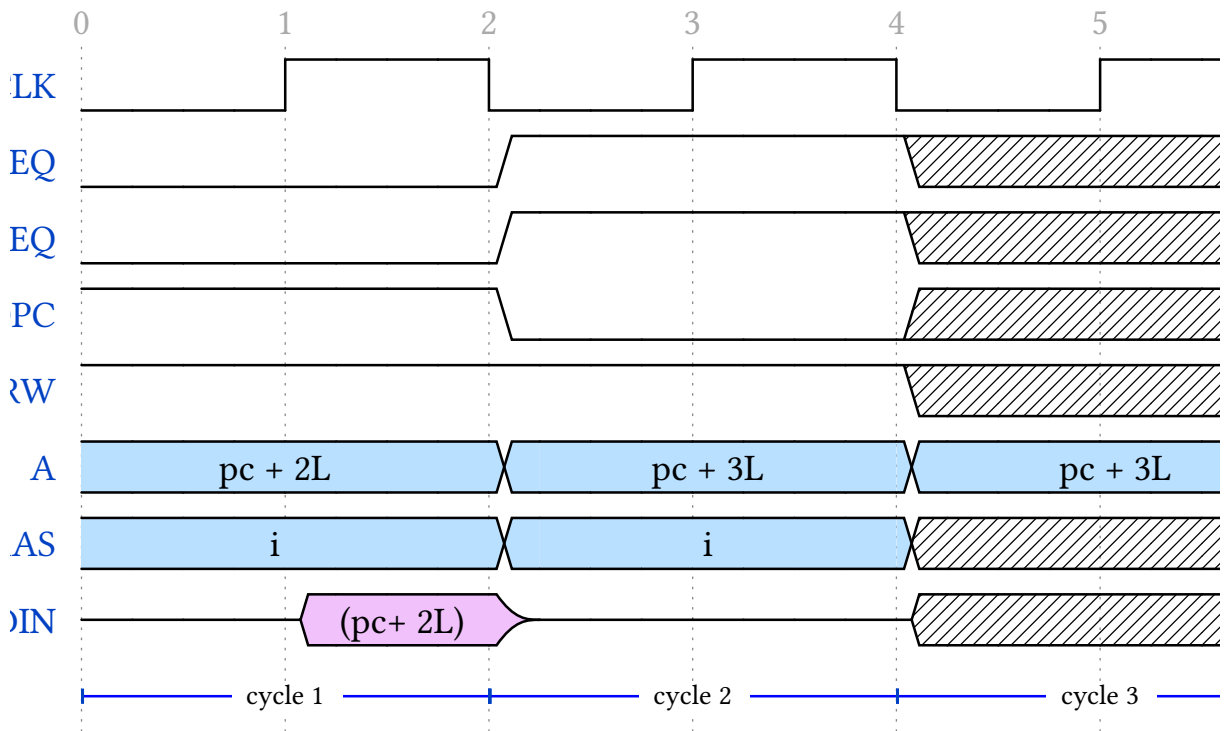


Figure 31: Data Processing Instruction Cycle (shift(RS))

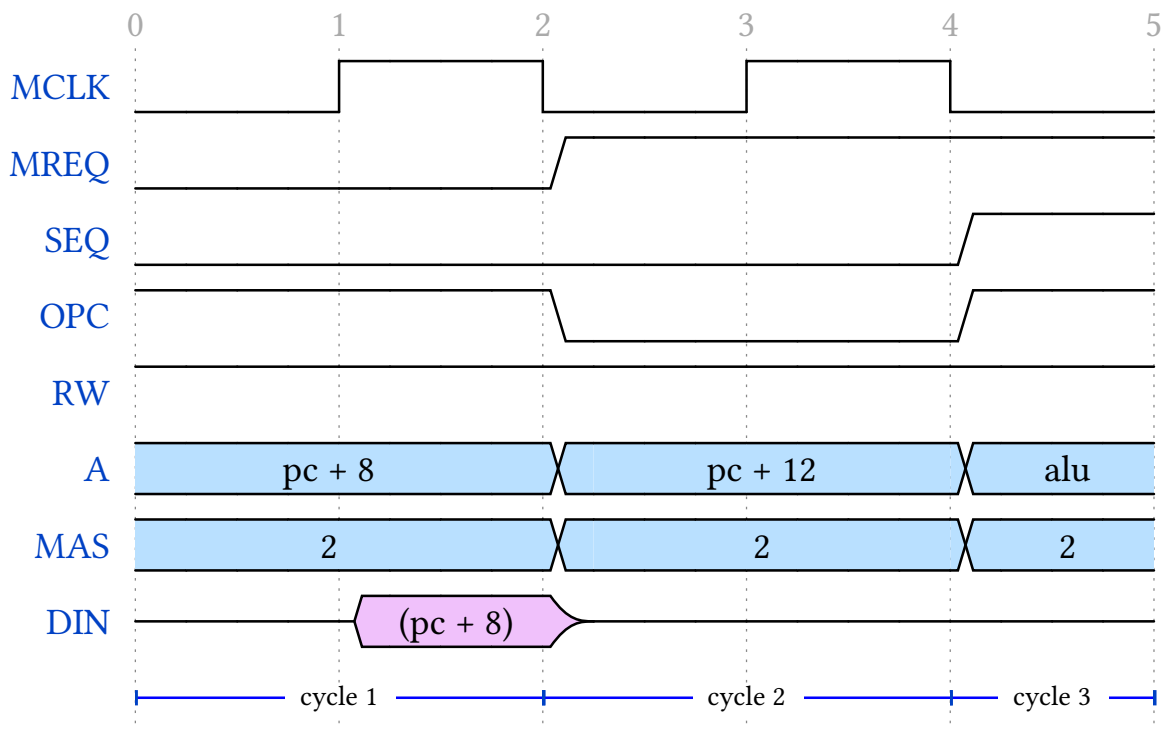


Figure 32: Data Processing Instruction Cycle (shift(Rs) dest=pc)

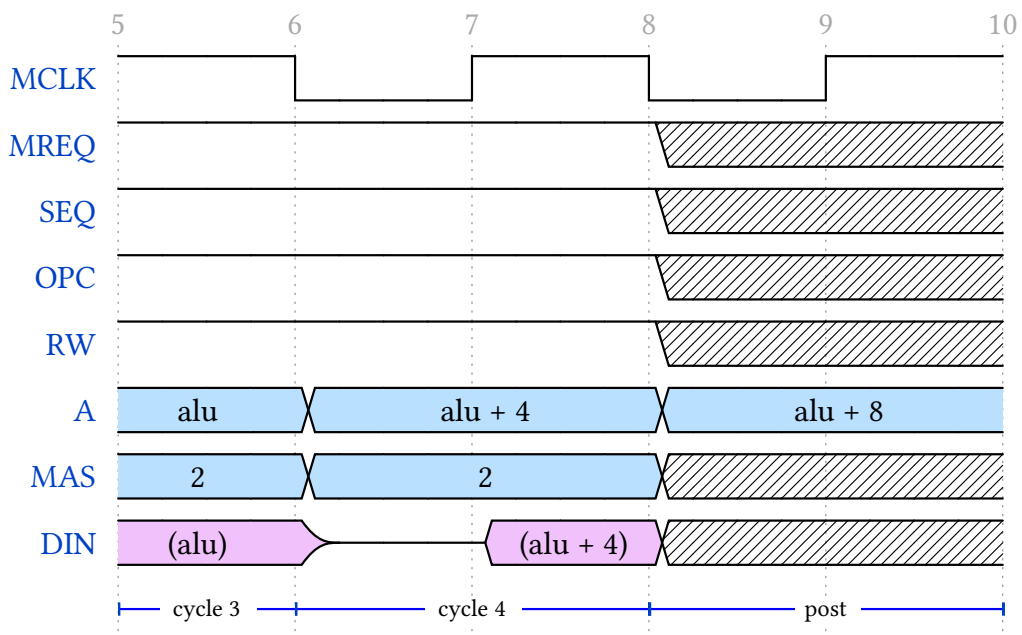


Figure 33: Data Processing Instruction Cycle (shift(Rs) dest=pc) (continued)

5.24. MULTIPLY AND MULTIPLY ACCUMULATE INSTRUCTION CYCLE

RELATED INSTRUCTIONS

MLA , MUL , SMLAL , SMULL , UMLAL , UMULL

RELATED PROCEDURES

test_multiply_and_multiply_accumulate_instruction_cycle Test procedure.

RELATED PROCEDURES	
gba_request_multiply_and_multiply_accumulate_instruction	Request procedure.
memory_respond_multiply_and_multiply_accumulate_instruction	Response procedure.

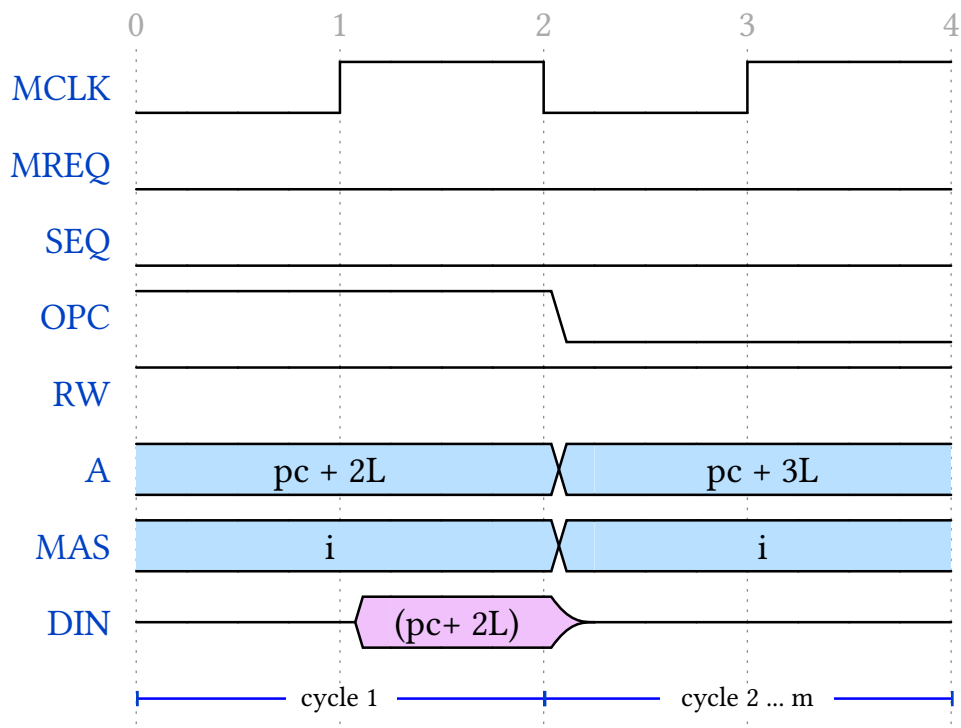


Figure 34: Multiply Instruction Cycle

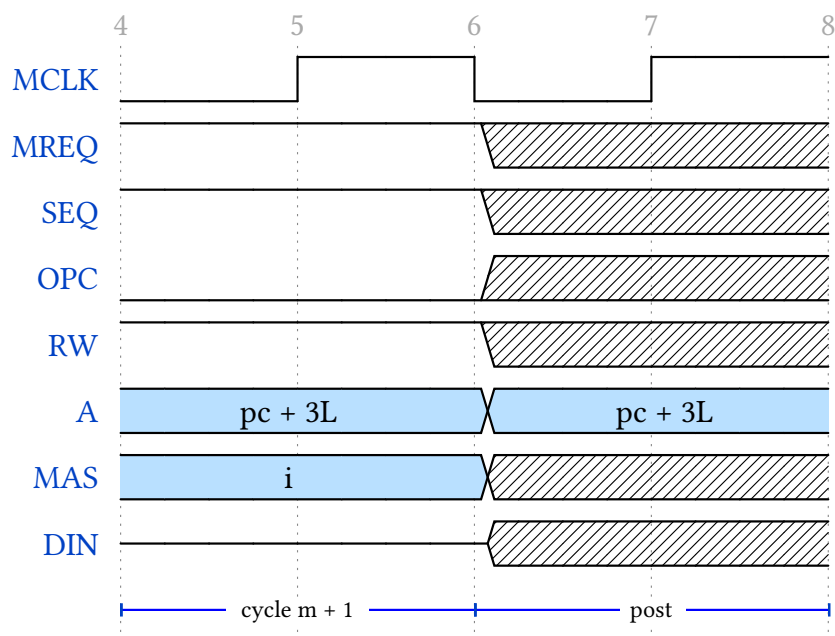


Figure 35: Multiply Instruction Cycle (continued)

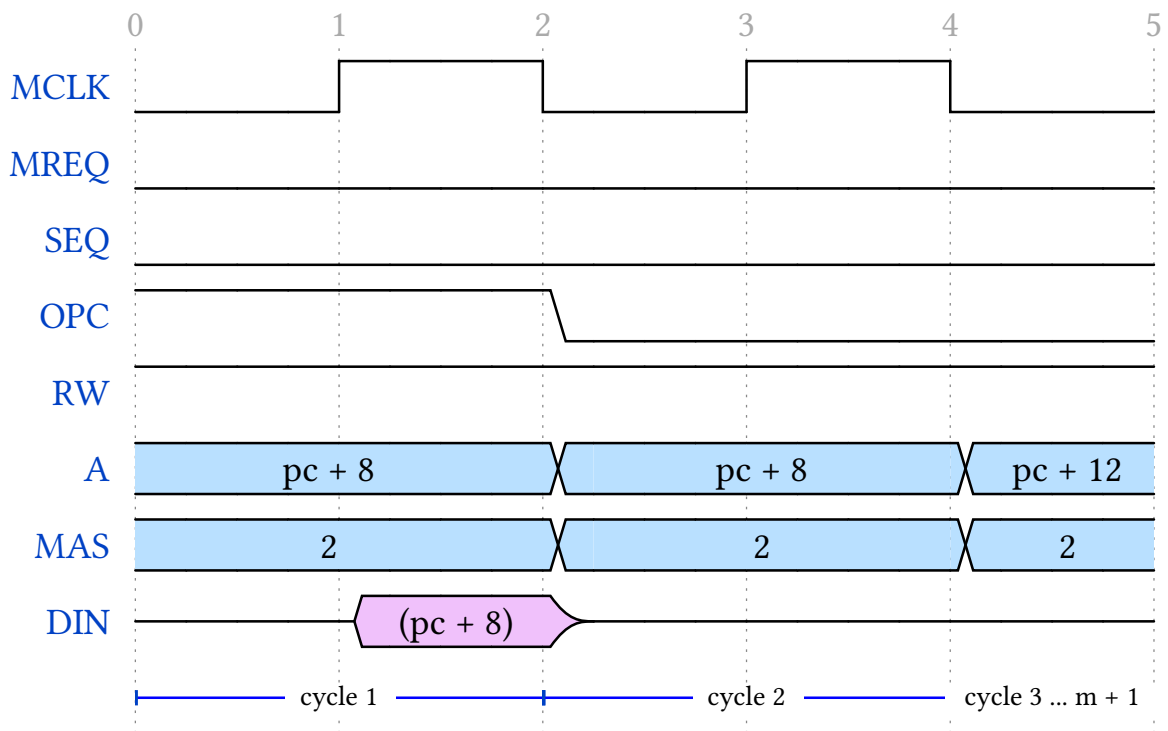


Figure 36: Multiply Accumulate Instruction Cycle

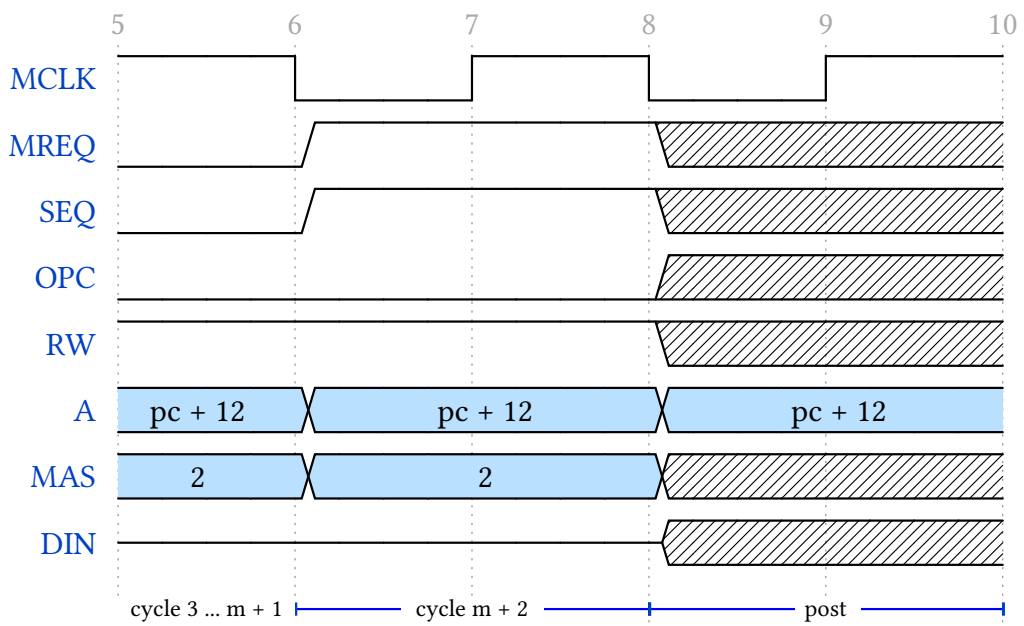


Figure 37: Multiply Accumulate Instruction Cycle (continued)

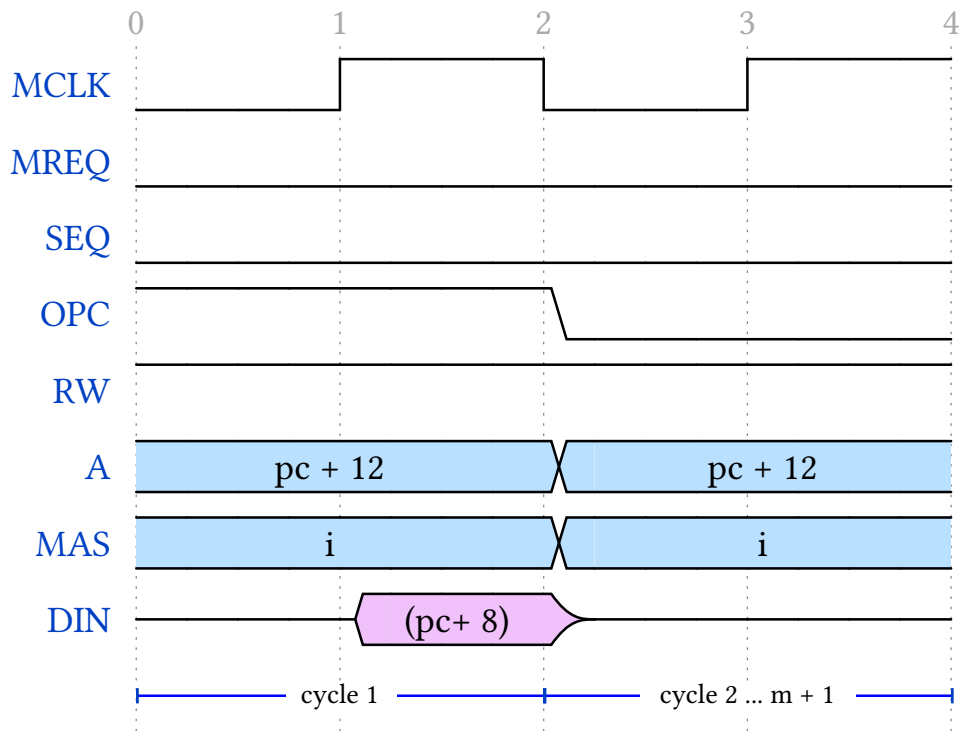


Figure 38: Multiply Long Instruction Cycle

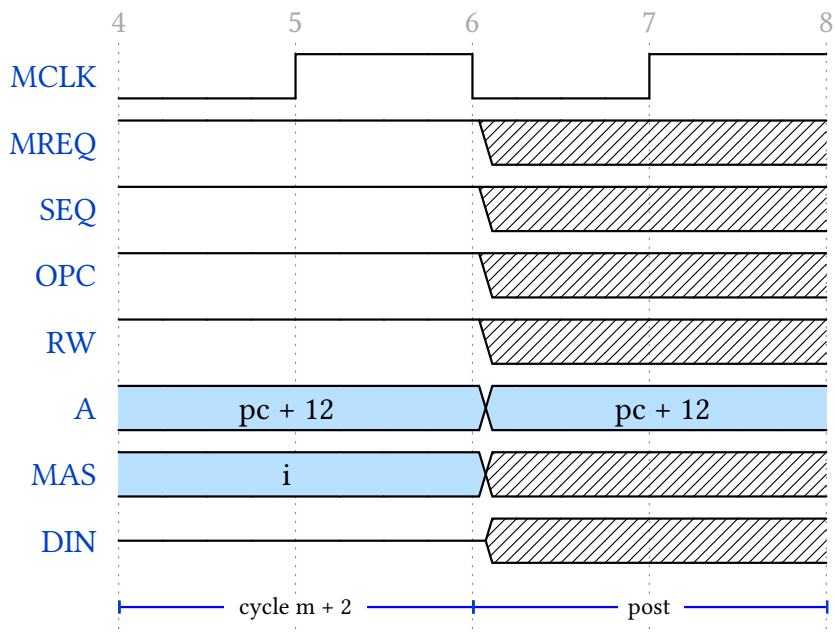


Figure 39: Multiply Long Instruction Cycle (continued)

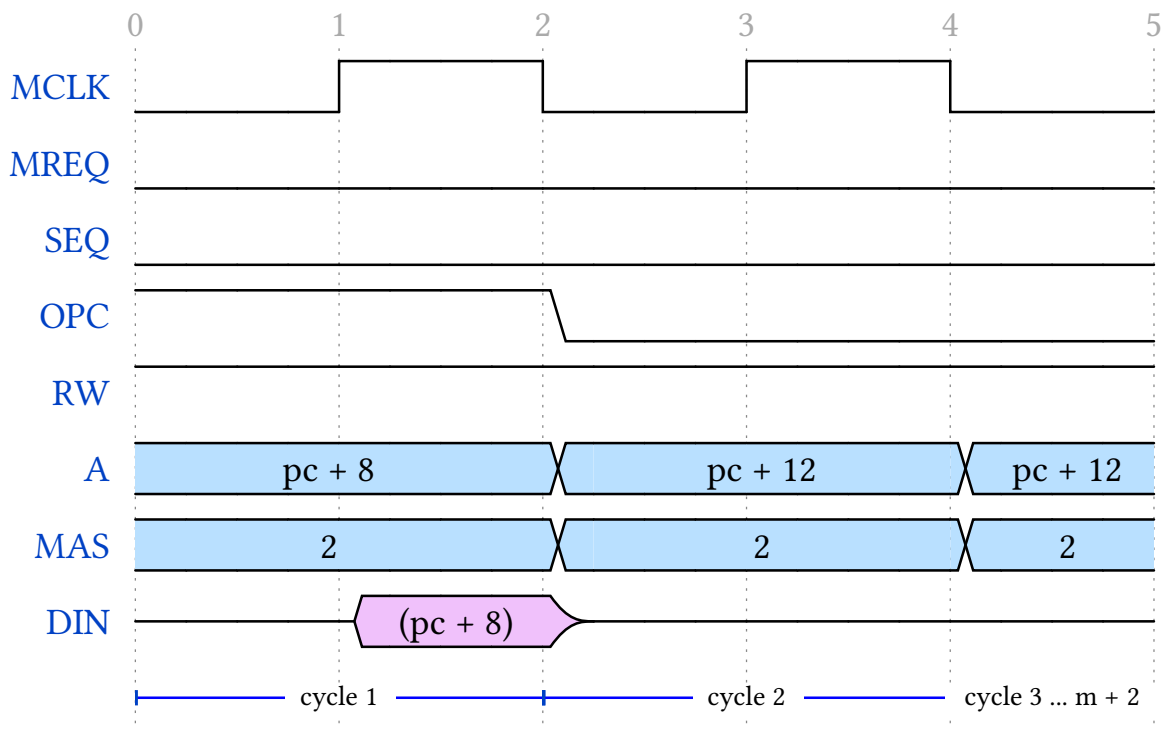


Figure 40: Multiply Accumulate Long Instruction Cycle

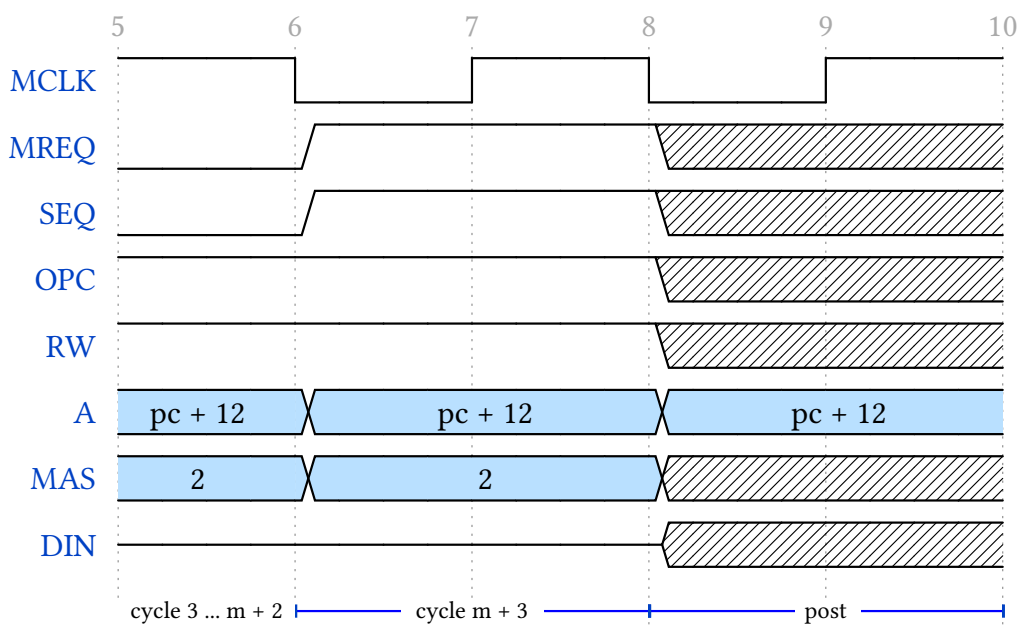


Figure 41: Multiply Accumulate Long Instruction Cycle (continued)

5.25. LOAD REGISTER INSTRUCTION CYCLE

RELATED INSTRUCTIONS

LDR , LDRB , LDRBT , LDRH , LDRSB , LDRSH , LDRT

RELATED PROCEDURES

test_load_register_instruction_cycle

Test procedure.

RELATED PROCEDURES	
gba_request_load_register_instruction_cycle	Request procedure.
memory_respond_load_register_instruction_cycle	Response procedure.

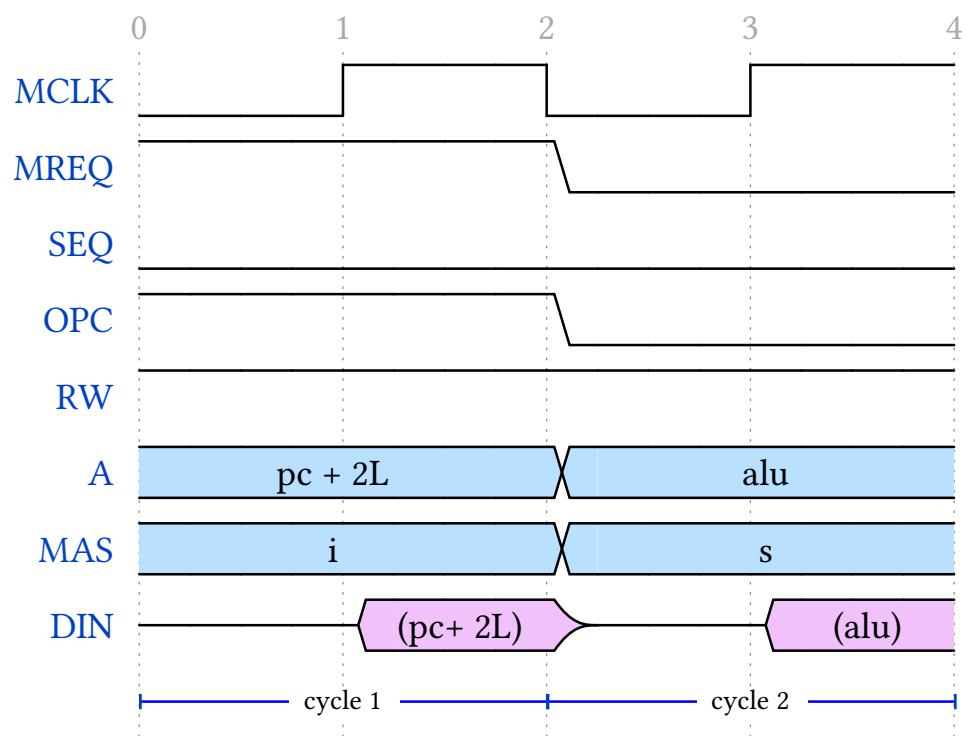


Figure 42: Load Register Instruction Cycle (normal)

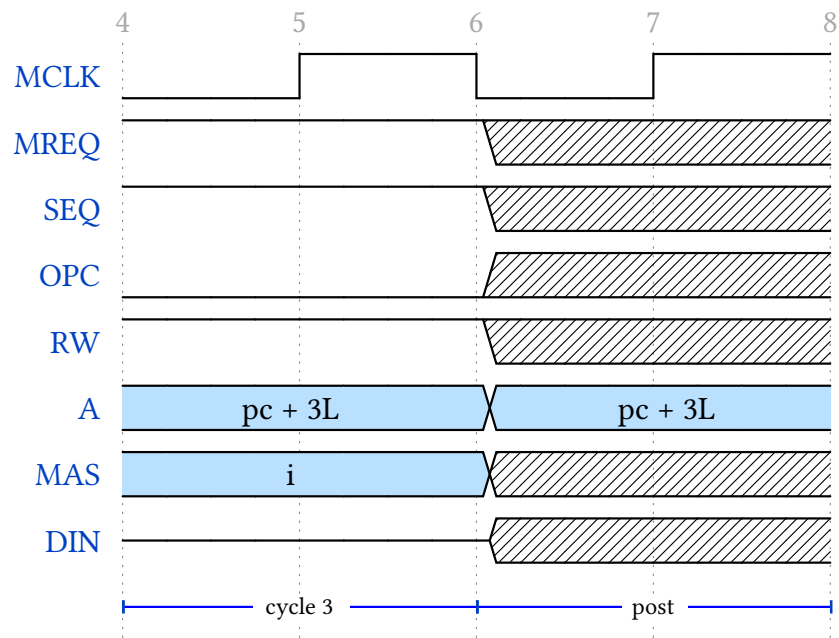


Figure 43: Load Register Instruction Cycle (normal) (continued)

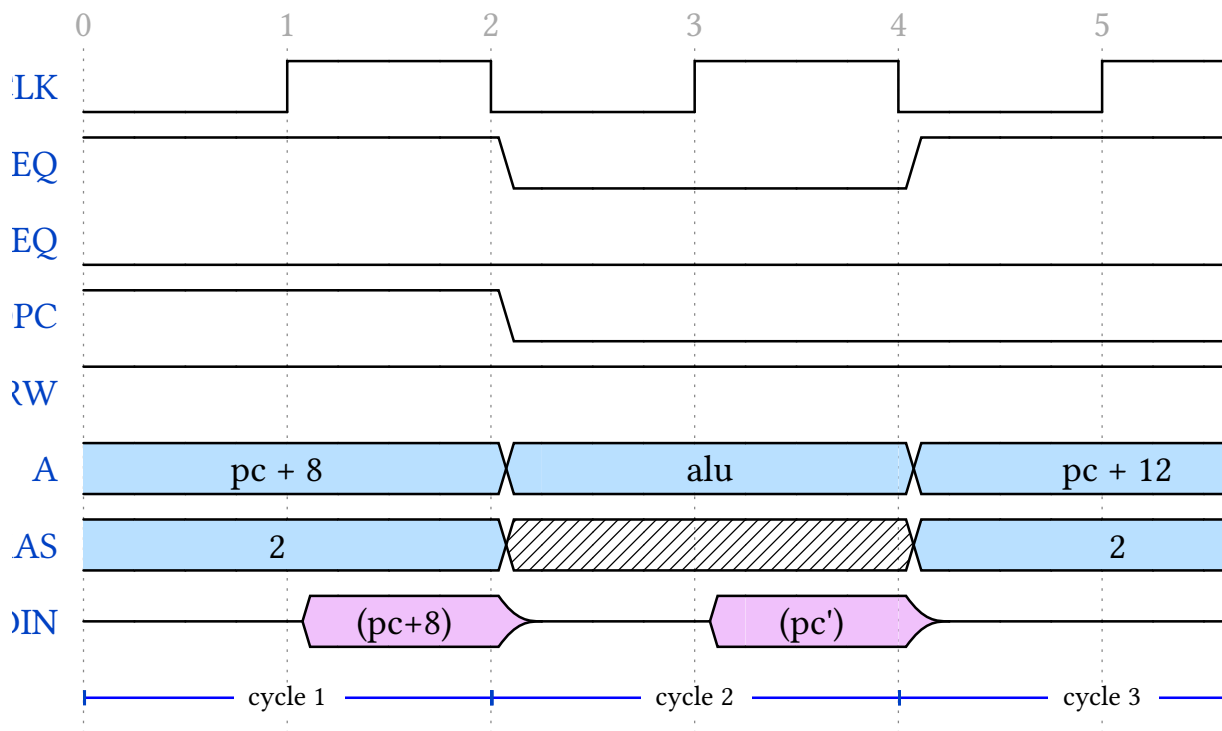


Figure 44: Load Register Instruction Cycle (dest=pc)

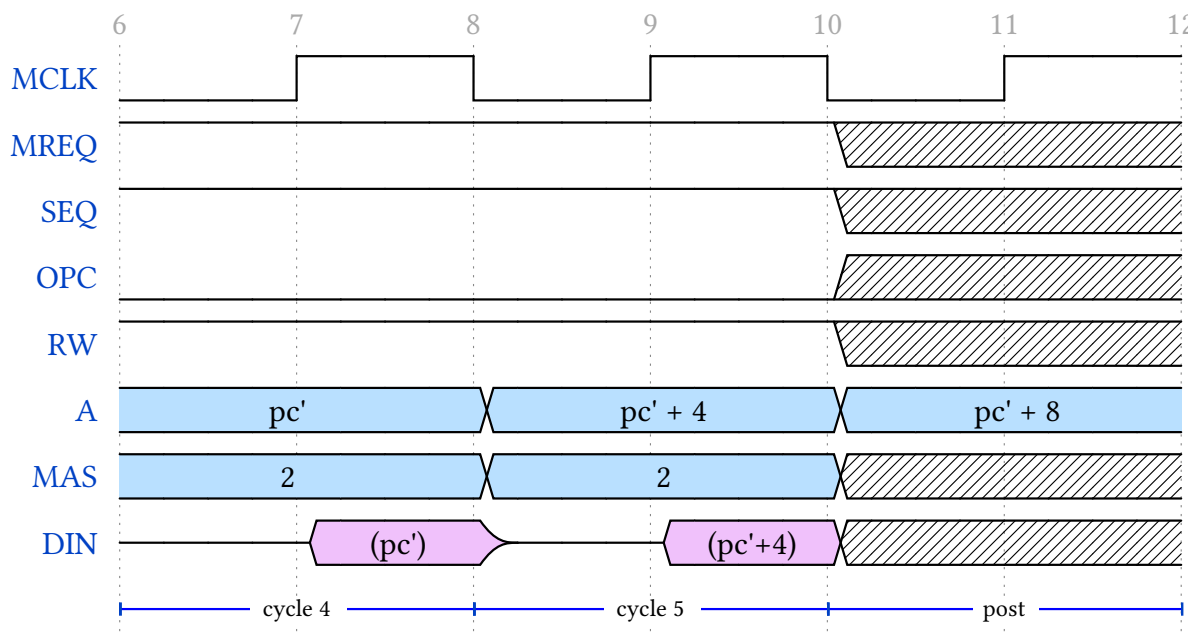


Figure 45: Load Register Instruction Cycle (dest=pc) (continued)

5.26. STORE REGISTER INSTRUCTION CYCLE

RELATED INSTRUCTIONS

STR , STRB , STRBT , STRH , STRT

RELATED PROCEDURES

test_store_register_instruction_cycle

Test procedure.

RELATED PROCEDURES	
gba_request_store_register_instruction_cycle	Request procedure.
memory_respond_store_register_instruction_cycle	Response procedure.

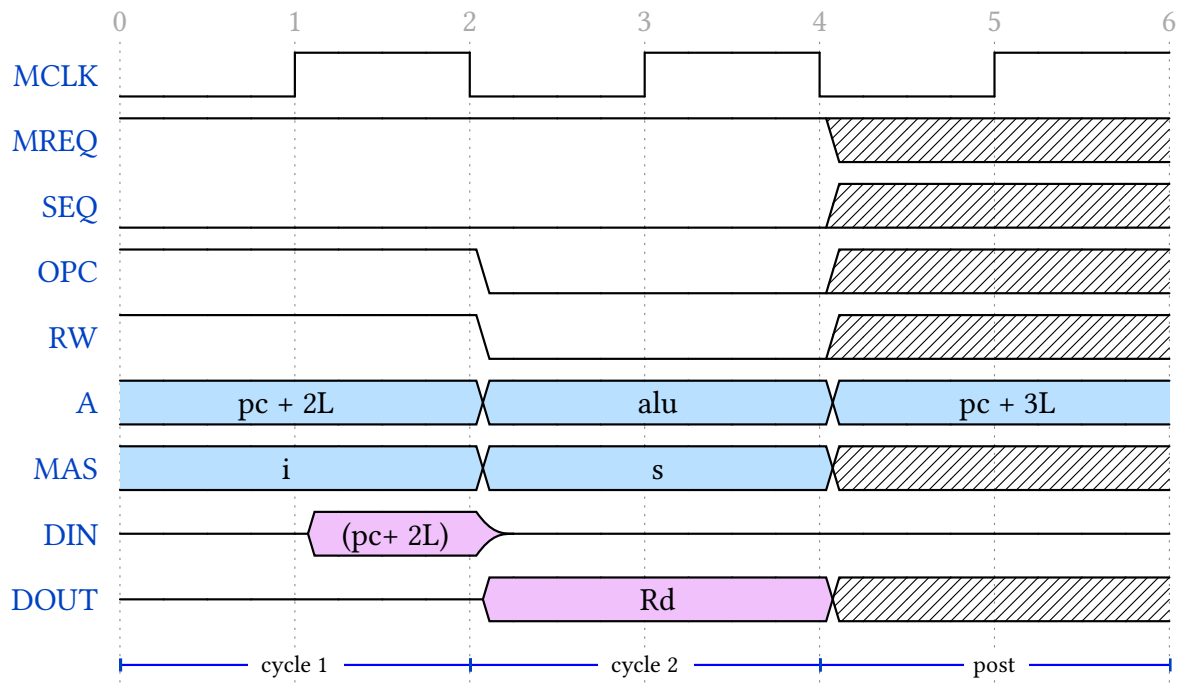


Figure 46: Store Register Instruction Cycle

5.27. LOAD MULTIPLE REGISTER INSTRUCTION CYCLE

RELATED INSTRUCTIONS	
LDM	

RELATED PROCEDURES	
test_load_multiple_register_instruction_cycle	Test procedure.
gba_request_load_multiple_register_instruction_cycle	Request procedure.
memory_respond_load_multiple_register_instruction_cycle	Response procedure.

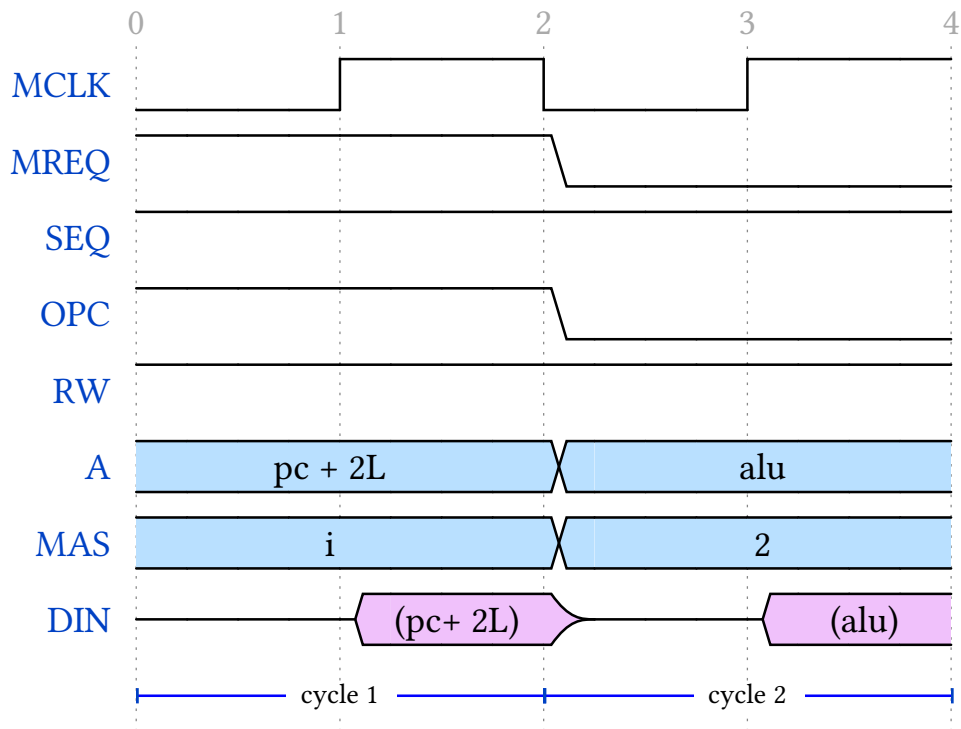


Figure 47: Load Multiple Register Instruction Cycle (single register)

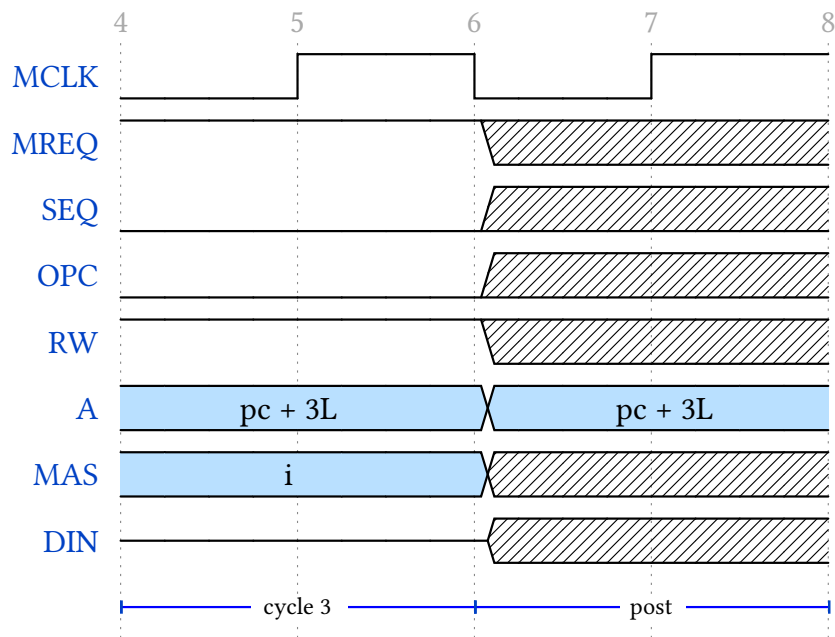


Figure 48: Load Multiple Register Instruction Cycle (single register) (continued)

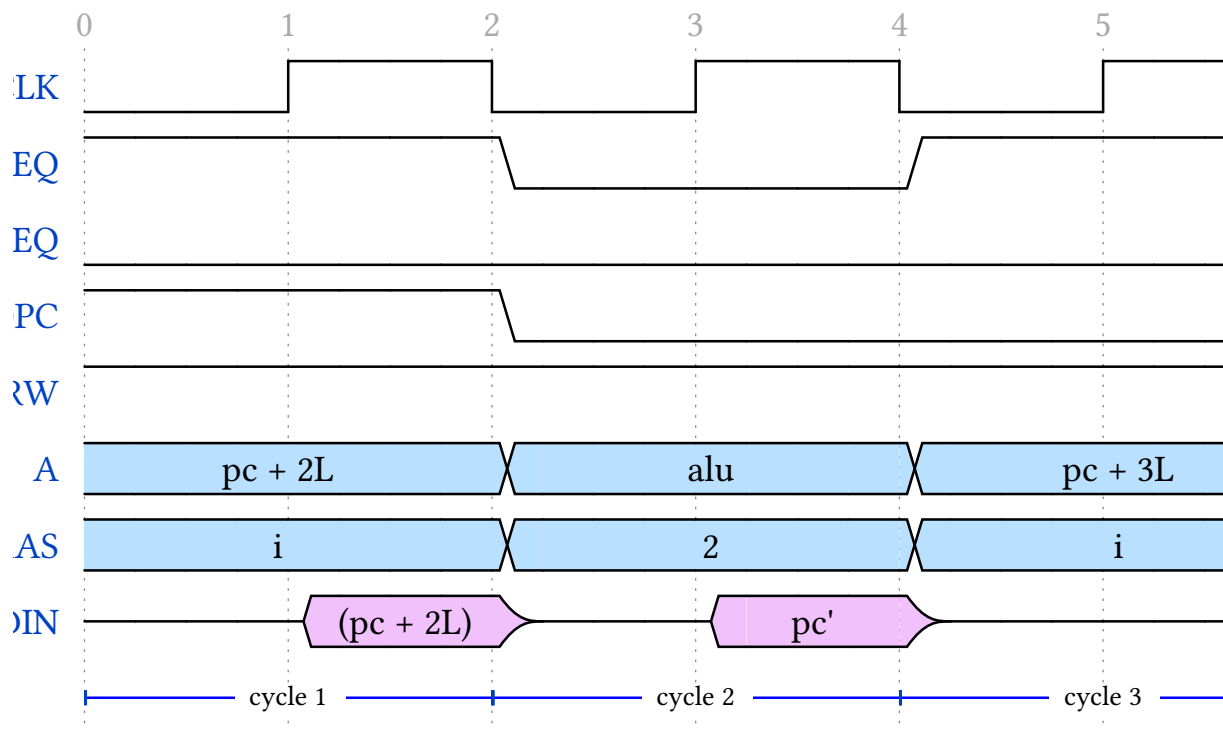


Figure 49: Load Multiple Register Instruction Cycle (single register dest=pc)

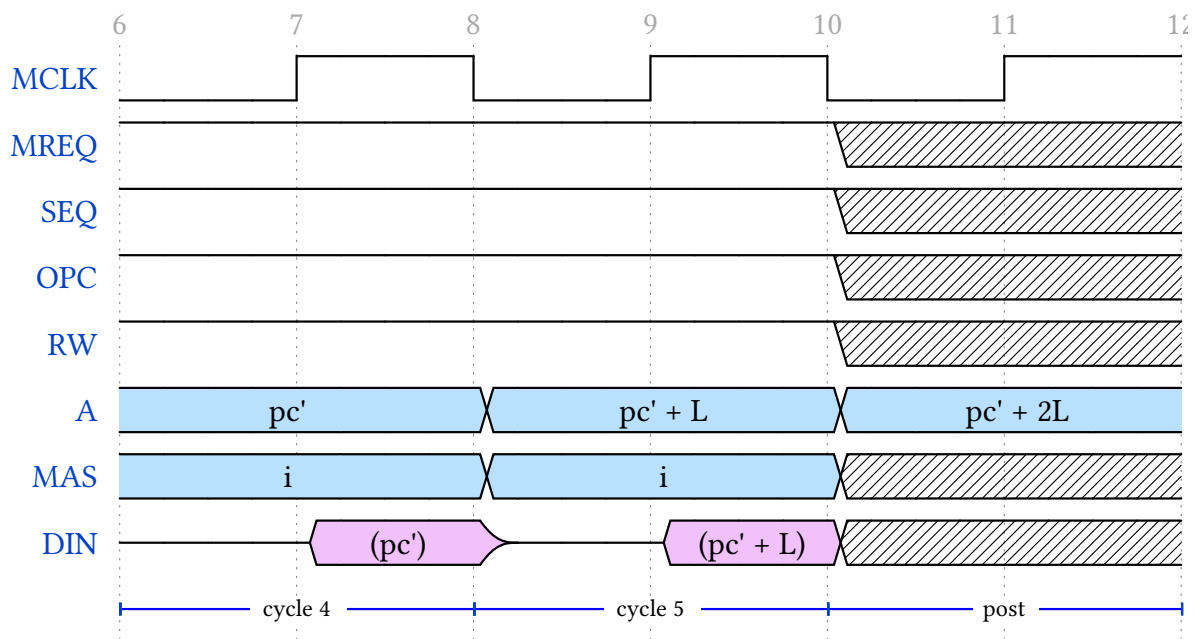


Figure 50: Load Multiple Register Instruction Cycle (single register dest=pc) (continued)

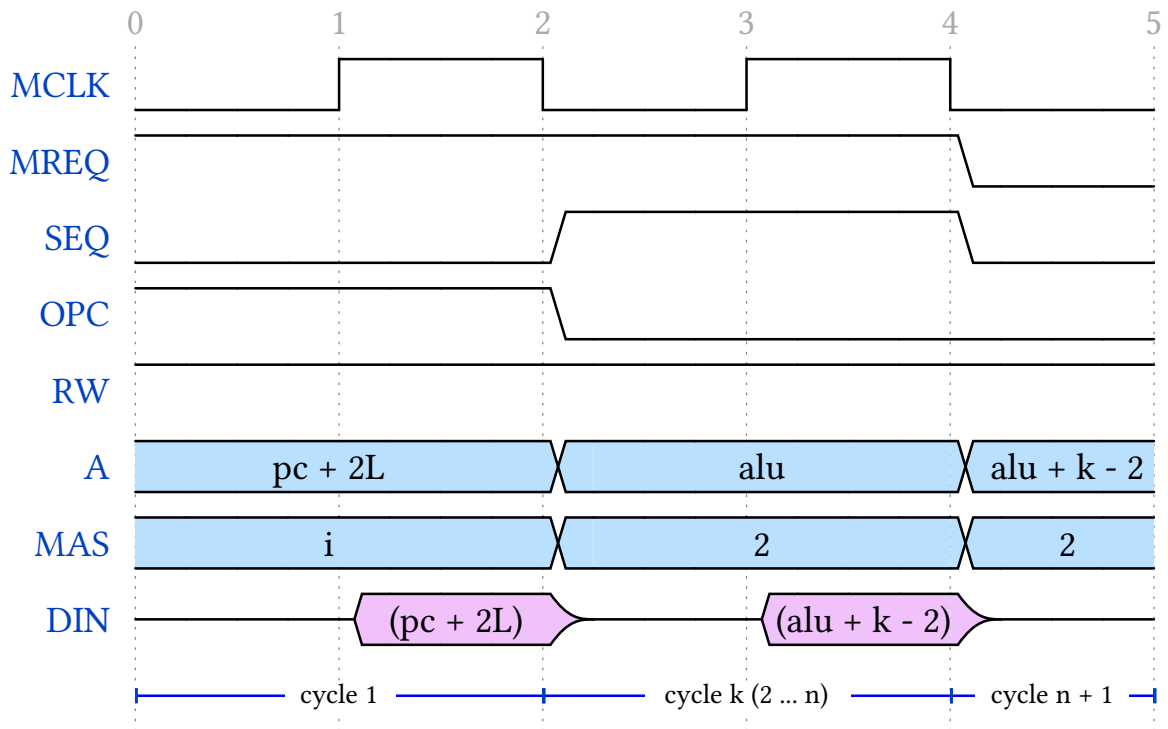


Figure 51: Load Multiple Register Instruction Cycle (n registers)

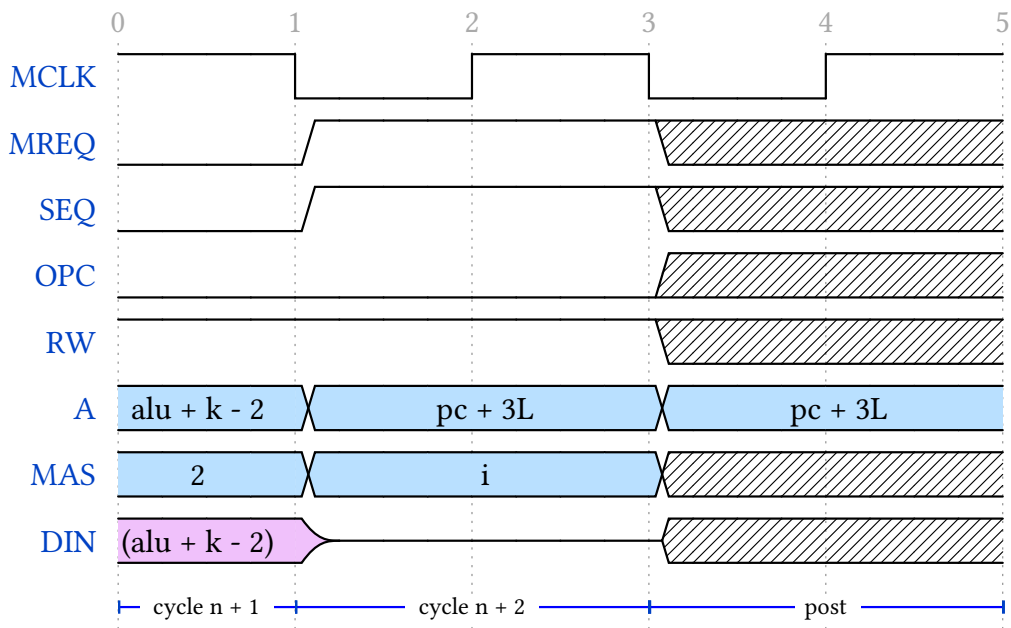


Figure 52: Load Multiple Register Instruction Cycle (n registers) (continued)

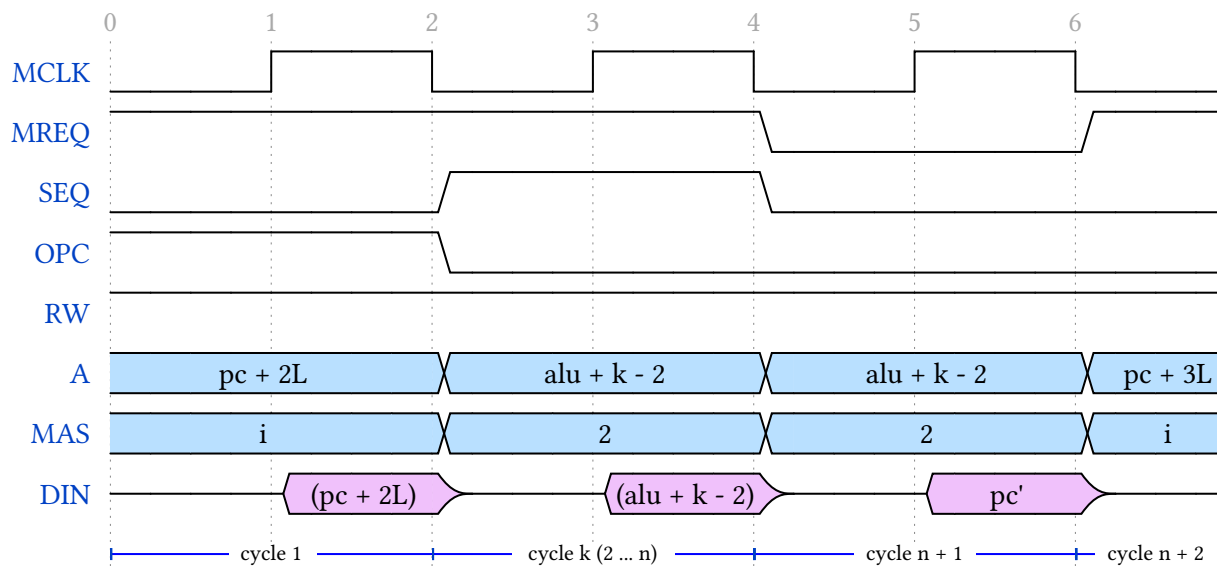


Figure 53: Load Multiple Register Instruction Cycle (n registers including pc)

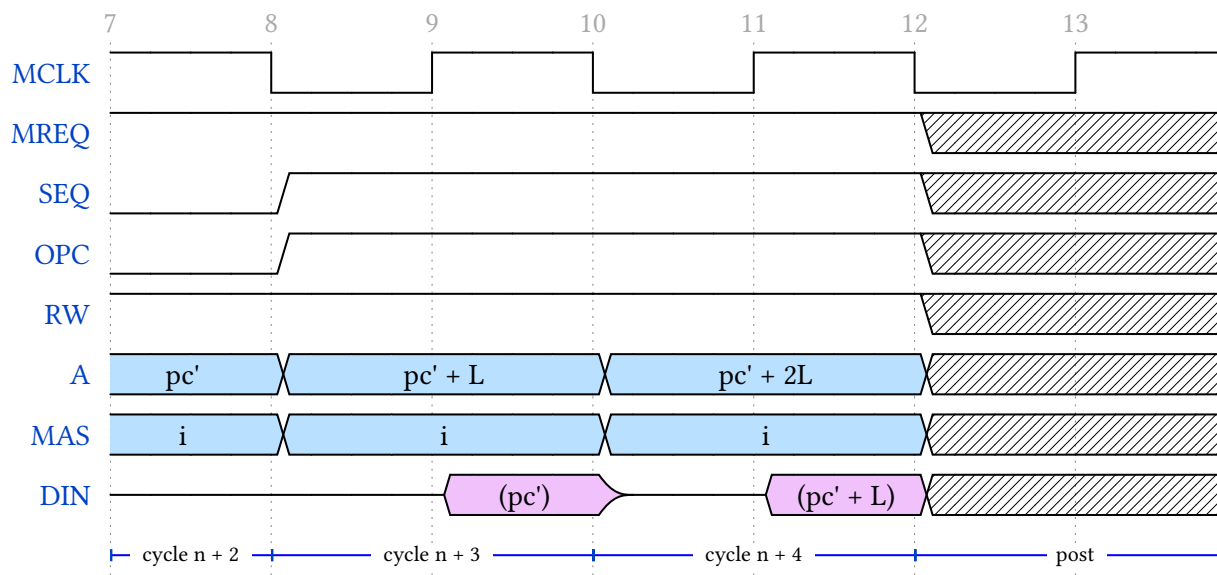


Figure 54: Load Multiple Register Instruction Cycle (n registers including pc) (continued)

5.28. STORE MULTIPLE REGISTER INSTRUCTION CYCLE

RELATED INSTRUCTIONS

STM

RELATED PROCEDURES

test_store_multiple_register_instruction_cycle	Test procedure.
gba_request_store_multiple_register_instruction_cycle	Request procedure.
memory_respond_store_multiple_register_instruction_cycle	Response procedure.

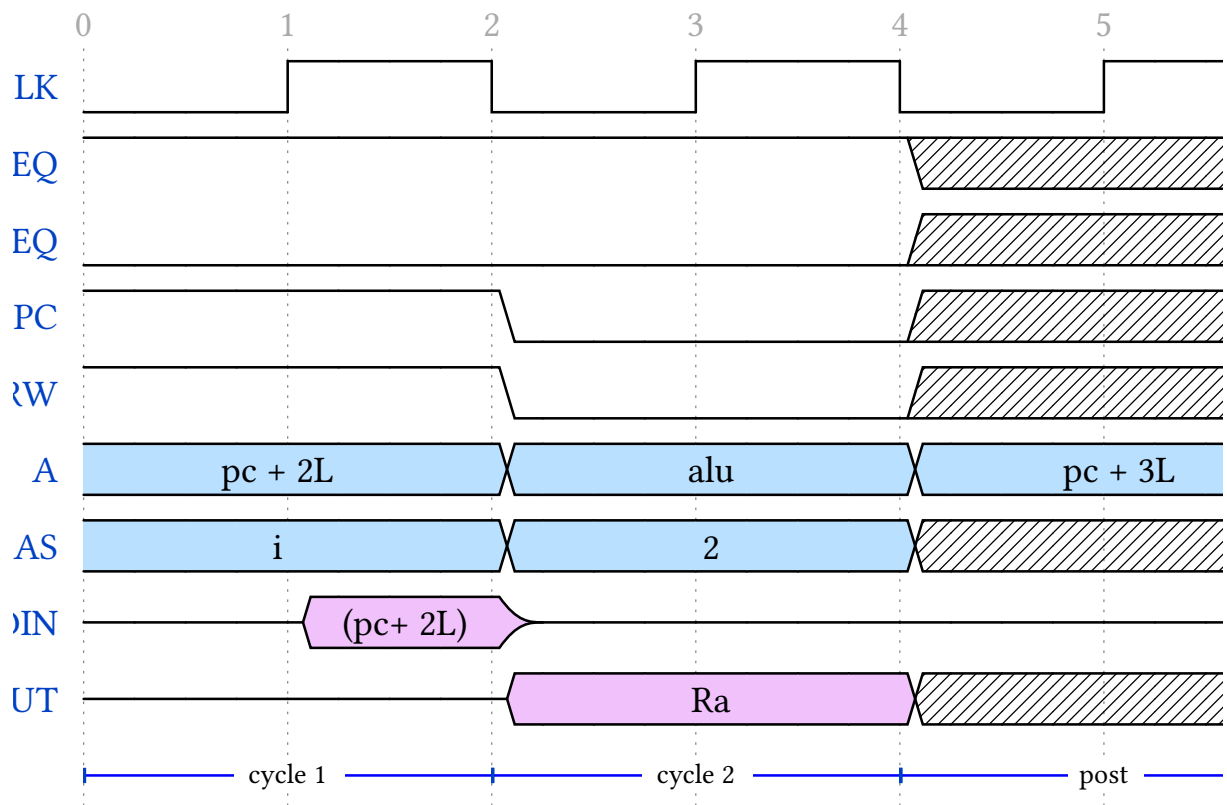


Figure 55: Store Multiple Register Instruction Cycle (single register)

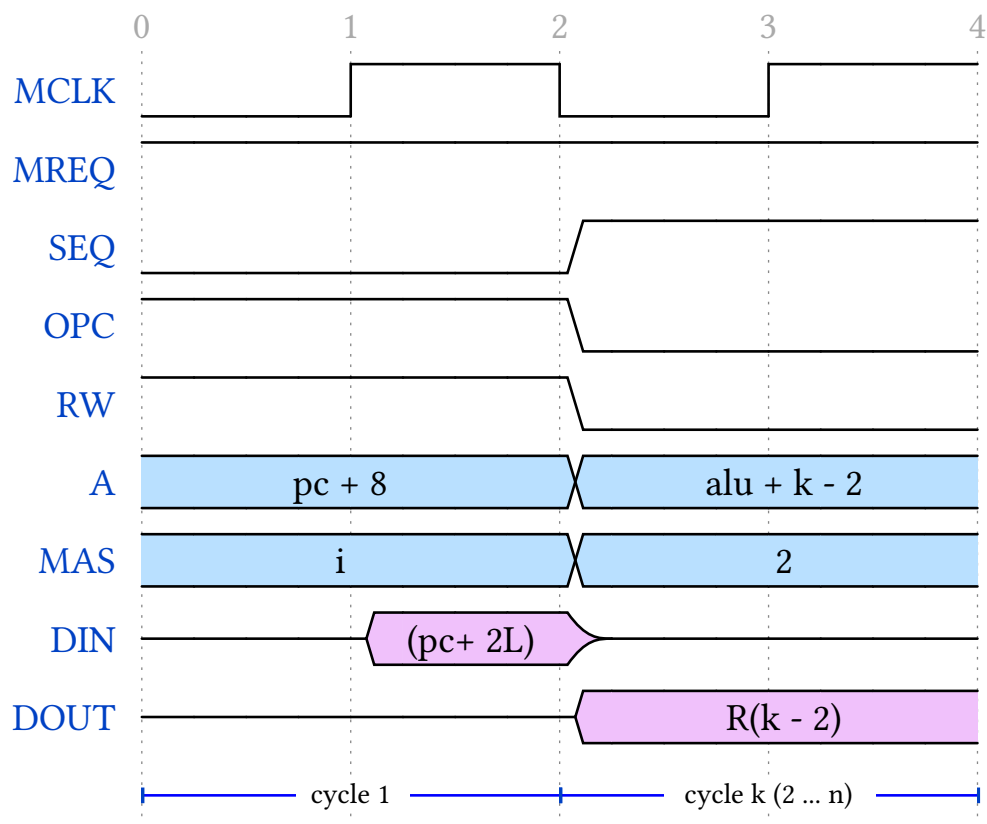


Figure 56: Store Multiple Register Instruction Cycle (n registers)

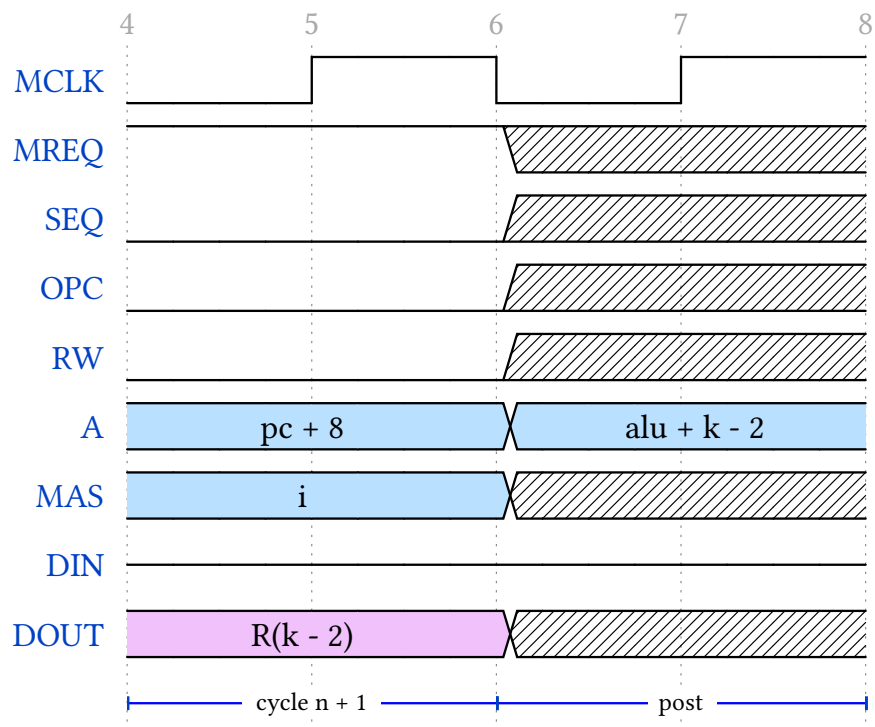


Figure 57: Store Multiple Register Instruction Cycle (n registers) (continued)

5.29. DATA SWAP INSTRUCTION CYCLE

RELATED INSTRUCTIONS

SWP , SWPB

RELATED PROCEDURES

test_data_swap_instruction_cycle	Test procedure.
gba_request_data_swap_instruction_cycle	Request procedure.
memory_respond_data_swap_instruction_cycle	Response procedure.

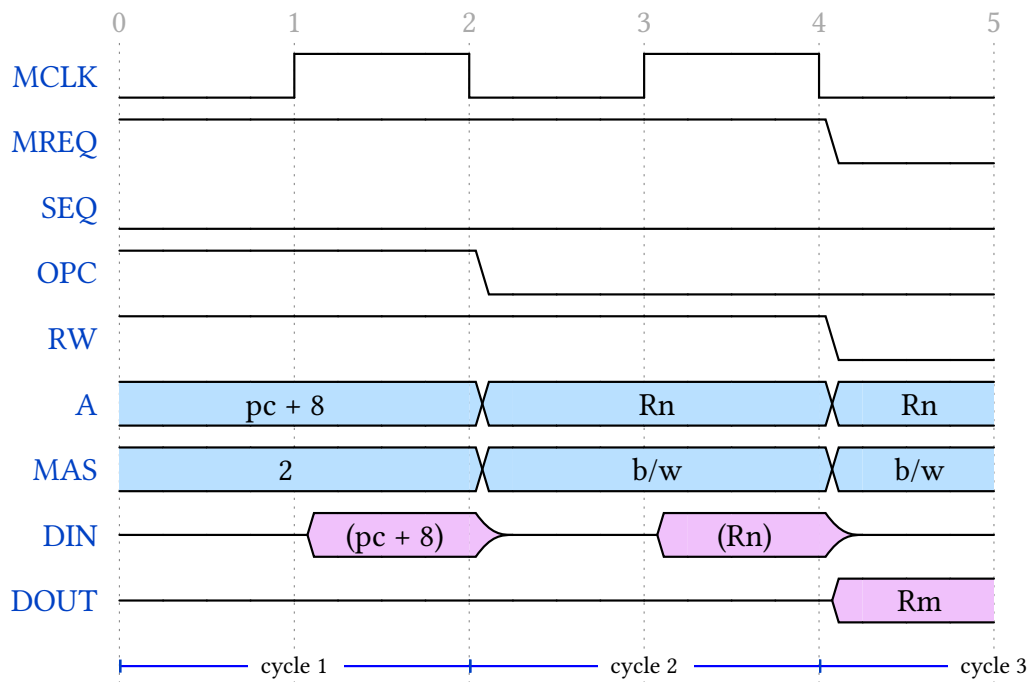


Figure 58: Data Swap Instruction Cycle

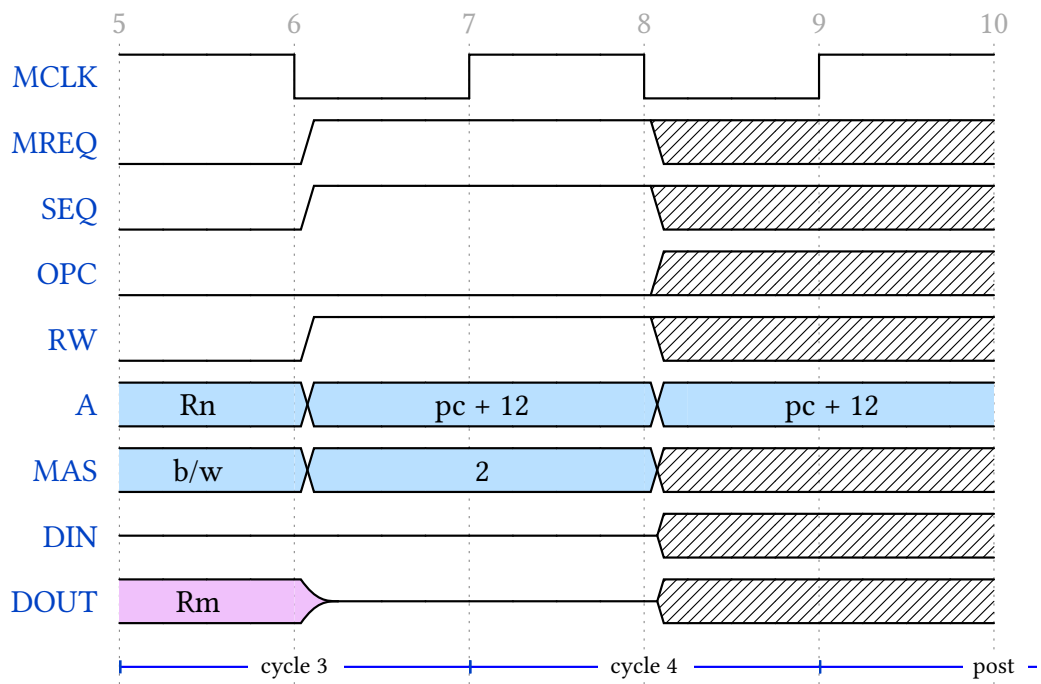


Figure 59: Data Swap Instruction Cycle (continued)

5.30. SOFTWARE INTERRUPT AND EXCEPTION INSTRUCTION CYCLE

RELATED INSTRUCTIONS

SWI

RELATED PROCEDURES

test_software_interrupt_and_exception_instruction_cycle Test procedure.

RELATED PROCEDURES	
gba_request_software_interrupt_and_exception_instruction	Request procedure.
memory_respond_software_interrupt_and_exception_instruction	Response procedure.

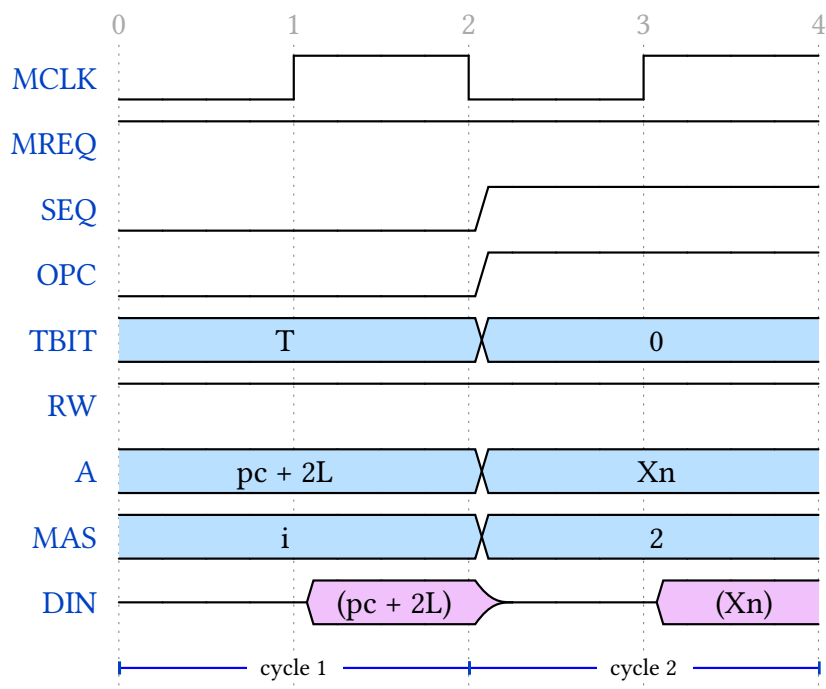


Figure 60: Software Interrupt and Exception Instruction Cycle

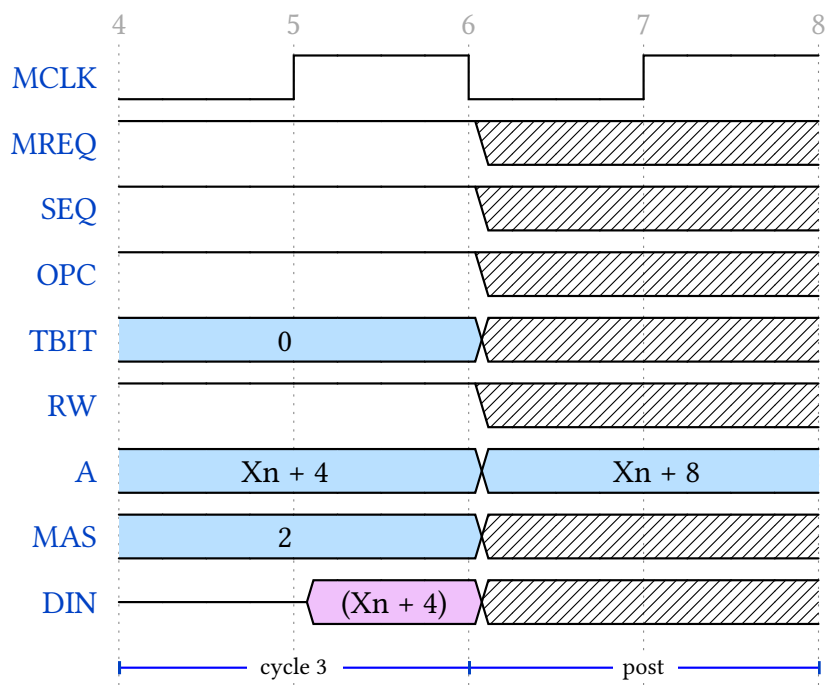


Figure 61: Software Interrupt and Exception Instruction Cycle (continued)

5.31. UNDEFINED INSTRUCTION CYCLE

RELATED PROCEDURES	
test_undefined_instruction_cycle	Test procedure.
gba_request_undefined_instruction_cycle	Request procedure.
memory_respond_undefined_instruction_cycle	Response procedure.

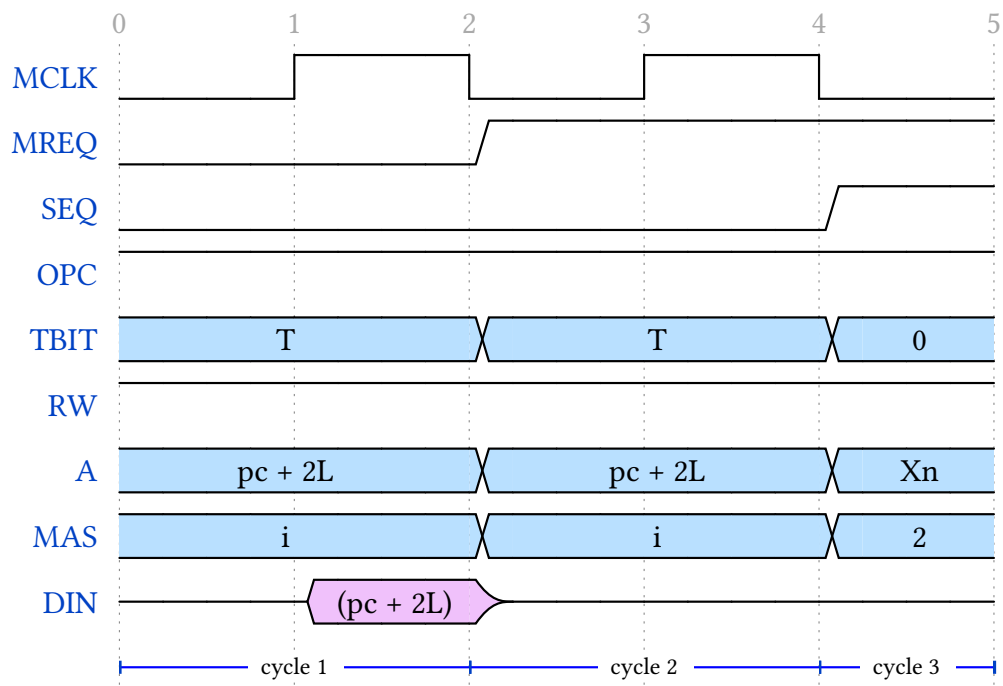


Figure 62: Undefined Instruction Cycle

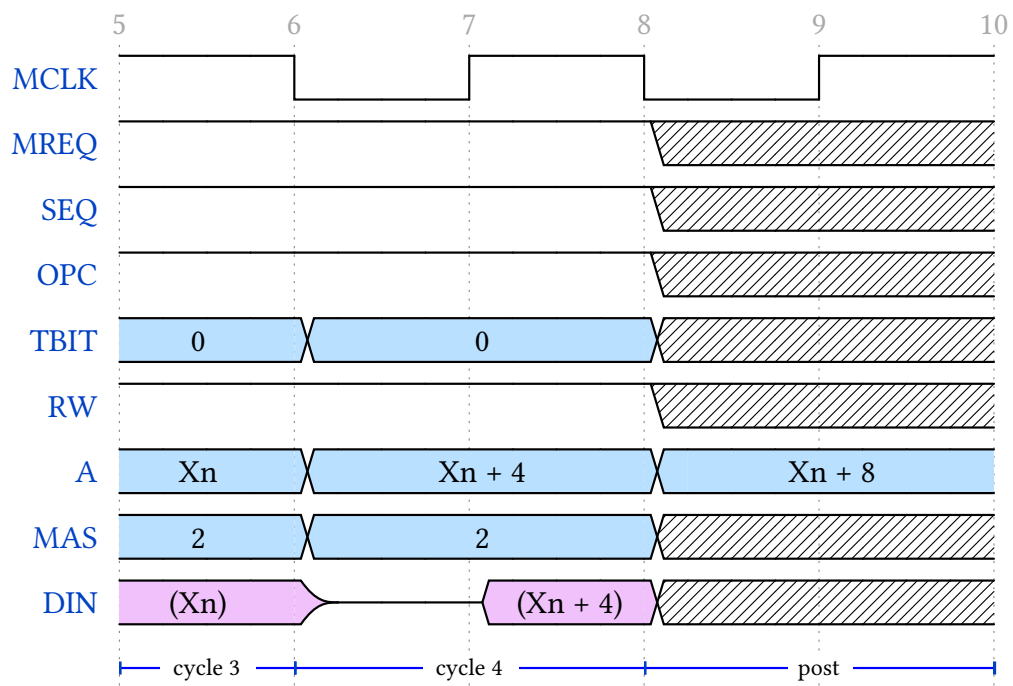


Figure 63: Undefined Instruction Cycle (continued)

5.32. UNEXECUTED INSTRUCTION CYCLE

RELATED PROCEDURES	
test_unexecuted_instruction_cycle	Test procedure.
gba_request_unexecuted_instruction_cycle	Request procedure.
memory_respond_unexecuted_instruction_cycle	Response procedure.

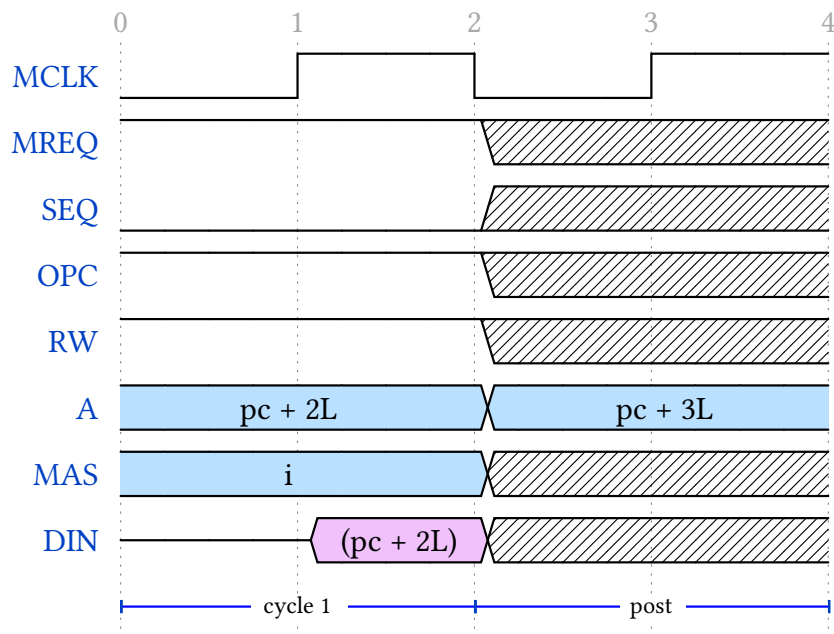


Figure 64: Unexecuted Instruction Cycle

Index of Figures

Figure 1	Memory Sequence timing diagram	9
Figure 2	Nonsequential Memory Sequence timing diagram	10
Figure 3	Sequential Memory Sequence timing diagram	11
Figure 4	Internal Sequence timing diagram	12
Figure 5	Merged Internal-Sequential Sequence timing diagram	12
Figure 6	Depipelined Addressing timing diagram	13
Figure 7	Data Write Sequence timing diagram	14
Figure 8	Data Read Sequence timing diagram	15
Figure 9	Halfword-Wide Read Memory Sequence timing diagram	16
Figure 10	Halfword-Wide Write Memory Sequence timing diagram	16
Figure 11	Byte-Wide Read Memory Sequence timing diagram	17
Figure 12	Byte-Wide Write Memory Sequence timing diagram	18
Figure 13	Reset Sequence timing diagram	19
Figure 14	Reset Sequence timing diagram (continued)	19
Figure 15	General timing diagram	20
Figure 16	Address Bus Control timing diagram	20
Figure 17	Address Bus Control timing diagram	21
Figure 18	Data Bus Control timing diagram	21
Figure 19	Exception Control timing diagram	22
Figure 20	Address Pipeline Control timing diagram	22
Figure 21	General Instruction Cycle timing diagram	23
Figure 22	Branch and Branch with Link Instruction Cycle	24
Figure 23	Branch and Branch with Link Instruction Cycle (continued)	24
Figure 24	Thumb Branch with Link Instruction Cycle	25
Figure 25	Thumb Branch with Link Instruction Cycle (continued)	25
Figure 26	Branch and Exchange Instruction Cycle	26
Figure 27	Branch and Exchange Instruction Cycle (continued)	26
Figure 28	Data Processing Instruction Cycle (normal)	27
Figure 29	Data Processing Instruction Cycle (dest=pc)	27
Figure 30	Data Processing Instruction Cycle (dest=pc) (continued)	28
Figure 31	Data Processing Instruction Cycle (shift(RS))	28
Figure 32	Data Processing Instruction Cycle (shift(Rs) dest=pc)	29
Figure 33	Data Processing Instruction Cycle (shift(Rs) dest=pc) (continued)	29
Figure 34	Multiply Instruction Cycle	30
Figure 35	Multiply Instruction Cycle (continued)	30
Figure 36	Multiply Accumulate Instruction Cycle	31
Figure 37	Multiply Accumulate Instruction Cycle (continued)	31
Figure 38	Multiply Long Instruction Cycle	32
Figure 39	Multiply Long Instruction Cycle (continued)	32
Figure 40	Multiply Accumulate Long Instruction Cycle	33
Figure 41	Multiply Accumulate Long Instruction Cycle (continued)	33
Figure 42	Load Register Instruction Cycle (normal)	34
Figure 43	Load Register Instruction Cycle (normal) (continued)	34

Figure 44 Load Register Instruction Cycle (dest=pc)	35
Figure 45 Load Register Instruction Cycle (dest=pc) (continued)	35
Figure 46 Store Register Instruction Cycle	36
Figure 47 Load Multiple Register Instruction Cycle (single register)	37
Figure 48 Load Multiple Register Instruction Cycle (single register) (continued)	37
Figure 49 Load Multiple Register Instruction Cycle (single register dest=pc)	38
Figure 50 Load Multiple Register Instruction Cycle (single register dest=pc) (continued) .	38
Figure 51 Load Multiple Register Instruction Cycle (n registers)	39
Figure 52 Load Multiple Register Instruction Cycle (n registers) (continued)	39
Figure 53 Load Multiple Register Instruction Cycle (n registers including pc)	40
Figure 54 Load Multiple Register Instruction Cycle (n registers including pc) (continued) .	40
Figure 55 Store Multiple Register Instruction Cycle (single register)	41
Figure 56 Store Multiple Register Instruction Cycle (n registers)	41
Figure 57 Store Multiple Register Instruction Cycle (n registers) (continued)	42
Figure 58 Data Swap Instruction Cycle	43
Figure 59 Data Swap Instruction Cycle (continued)	43
Figure 60 Software Interrupt and Exception Instruction Cycle	44
Figure 61 Software Interrupt and Exception Instruction Cycle (continued)	44
Figure 62 Undefined Instruction Cycle	45
Figure 63 Undefined Instruction Cycle (continued)	45
Figure 64 Unexecuted Instruction Cycle	46