

# **GBANA DESIGN DOC**

Version 0.1.0

April 25, 2025

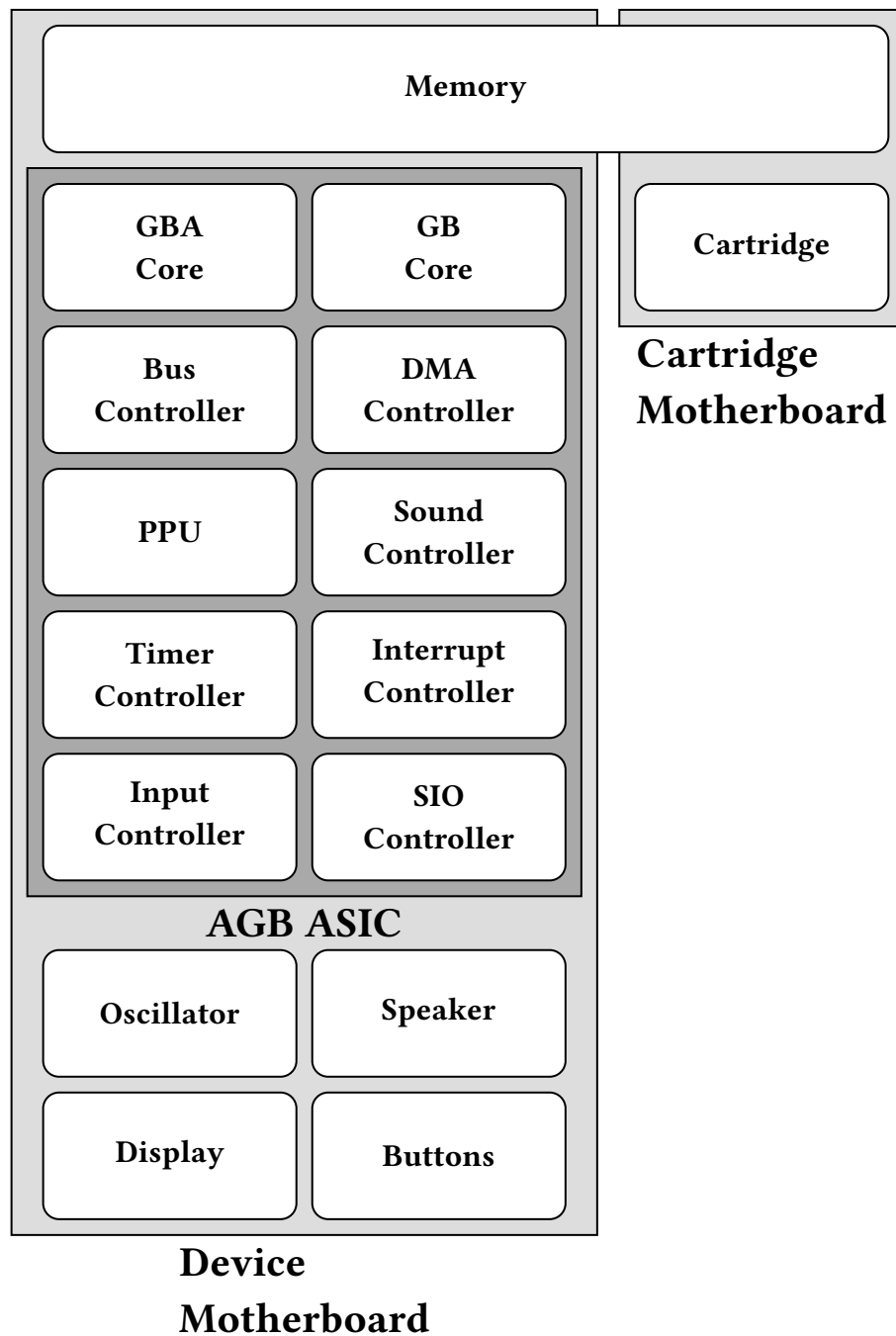
# Contents

1. CLASSES .....	3
2. BLOCK DIAGRAM .....	4
3. EMULATING THE CLOCK & CYCLE .....	5
4. SIGNALS .....	6
5. SEQUENCES .....	8
5.1. MEMORY SEQUENCE .....	9
5.2. NONSEQUENTIAL MEMORY SEQUENCE (N-CYCLE) .....	10
5.3. SEQUENTIAL MEMORY SEQUENCE (S-CYCLE) .....	11
5.4. INTERNAL SEQUENCE .....	12
5.5. MERGED INTERNAL-SEQUENTIAL SEQUENCE (MIS-CYCLE) .....	13
5.6. DEPIPELINED ADDRESSING .....	14
5.7. DATA WRITE SEQUENCE (DW-CYCLE) .....	15
5.8. DATA READ SEQUENCE (DR-CYCLE) .....	16
5.9. HALFWORD-WIDE MEMORY SEQUENCE .....	17
5.10. BYTE-WIDE MEMORY SEQUENCE .....	19
5.11. RESET SEQUENCE (RS-CYCLE) .....	21
5.12. GENERAL TIMING .....	22
5.13. ADDRESS BUS CONTROL .....	23
5.14. DATA BUS CONTROL .....	24
5.15. EXCEPTION CONTROL .....	25
5.16. ADDRESS PIPELINE CONTROL .....	26
5.17. GENERAL INSTRUCTION CYCLE .....	27
5.18. BRANCH AND BRANCH WITH LINK INSTRUCTION CYCLE (BABLI-CYCLE) ....	28
5.19. THUMB BRANCH WITH LINK INSTRUCTION CYCLE (TBLI-CYCLE) .....	30
5.20. BRANCH AND EXCHANGE INSTRUCTION CYCLE (BAEI-CYCLE) .....	32
5.21. DATA PROCESSING INSTRUCTION CYCLE (DPI-CYCLE) .....	34
5.22. MULTIPLY AND MULTIPLY ACCUMULATE INSTRUCTION CYCLE (MAMAI- CYCLE) .....	38
5.23. LOAD REGISTER INSTRUCTION CYCLE (LRI-CYCLE) .....	43
5.24. STORE REGISTER INSTRUCTION CYCLE (SRI-CYCLE) .....	46
5.25. LOAD MULTIPLE REGISTER INSTRUCTION CYCLE (LMRI-CYCLE) .....	47
5.26. STORE MULTIPLE REGISTER INSTRUCTION CYCLE (SMRI-CYCLE) .....	52
5.27. DATA SWAP INSTRUCTION CYCLE (DSI-CYCLE) .....	54
5.28. SOFTWARE INTERRUPT AND EXCEPTION INSTRUCTION CYCLE (SIAEI- CYCLE) .....	56
5.29. UNDEFINED INSTRUCTION CYCLE (UDI-CYCLE) .....	58
5.30. UNEXECUTED INSTRUCTION CYCLE (UEI-CYCLE) .....	60
Index of Figures .....	61

## 1. CLASSES

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

## 2. BLOCK DIAGRAM



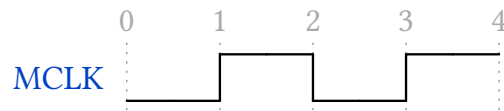
Each rounded rectangle is a **component**. Each component has a state object, which holds its data. Each non-rounded rectangle is a group of components that are semantically related. Components operate concurrently and share data only by means of **signals**.

### 3. EMULATING THE CLOCK & CYCLE

#### Related procedures:

`test_main_clock` Test procedure.

The core will initially be written as phase-accurate (every phase of every clock cycle is simulated), whose logic will be correct at phase bounds. Then a more high-level core will be implemented, which will be sequence-accurate, whose logic will be correct at sequence bounds, and this core will be verified against the lower-level core.



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

NAME	CLASS	COMPONENTS	DESCRIPTION
LOCK	Memory Interface	<b>Memory / Bus Controller</b>	The <b>GBA Core / GB Core</b> sets this to high to gain exclusive access to the <b>Memory</b> (this is effectively a mutex on the <b>Memory</b> signals).
MAS	Memory Interface	<b>Memory / Bus Controller</b>	(Memory Access Size) The <b>GBA Core / GB Core</b> writes here the size of the requested data access.
MCLK	Clocks and Timing	<b>GBA Core, Memory</b>	(Main Clock) The main clock.
M	Processor Mode	<b>GBA Core</b>	(Processor Mode) The current mode of the ARM7TDMI core.
MREQ	Memory Interface	<b>Memory / Bus Controller</b>	(Memory Request) The <b>GBA Core / GB Core</b> sets this to high to request memory access in the subsequent cycle.
OPC	Memory Interface	<b>Memory</b>	(Op-Code Fetch) The <b>GBA Core / GB Core</b> sets this to high to indicate that the requested memory access is to fetch the next instruction.
RESET	Bus Controls	<b>GBA Core</b>	Set this to high to initialize / restart the ARM7TDMI processor.
RW	Memory Interface	<b>Memory / Bus Controller</b>	(Read/Write) The <b>GBA Core / GB Core</b> 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	<b>Memory / Bus Controller</b>	(Sequential Cycle) The <b>GBA Core / GB Core</b> 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	<b>GBA Core</b>	(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	<b>GBA Core</b>	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	
test_memory_sequence	
gba_request_memory_sequence	
memory_respond_memory_sequence	

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

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.

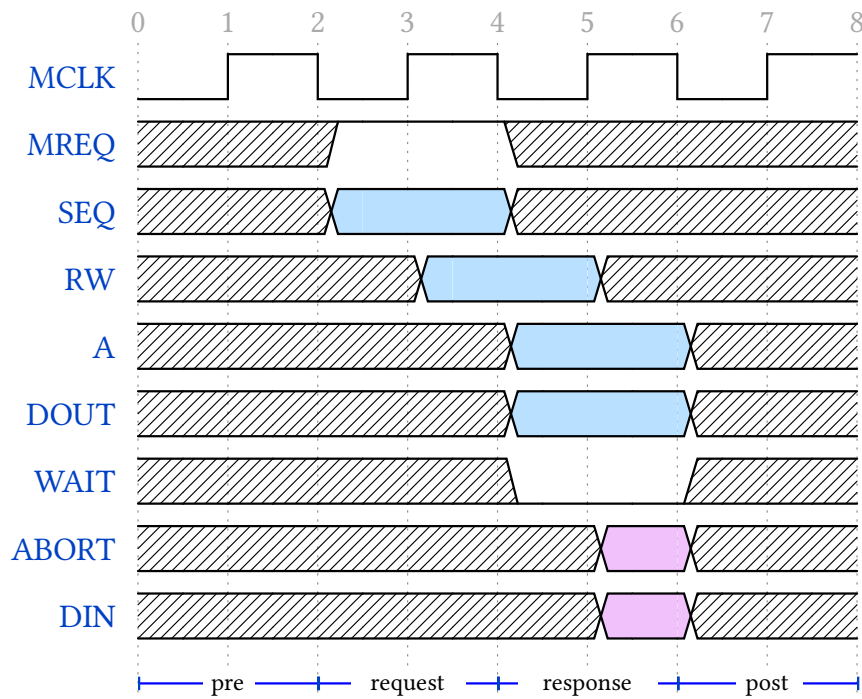


Figure 1: Memory Sequence timing diagram

- 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.

## 5.2. NONSEQUENTIAL MEMORY SEQUENCE (N-CYCLE)

RELATED PROCEDURES	
test_N_cycle	
gba_request_N_cycle	
memory_respond_N_cycle	
REQUEST SIGNALS	RESPONSE SIGNALS
MREQ , SEQ , RW , A , DOUT	WAIT , ABORT , DIN
DURATION: 1 TO 2 CLOCK CYCLES	

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.

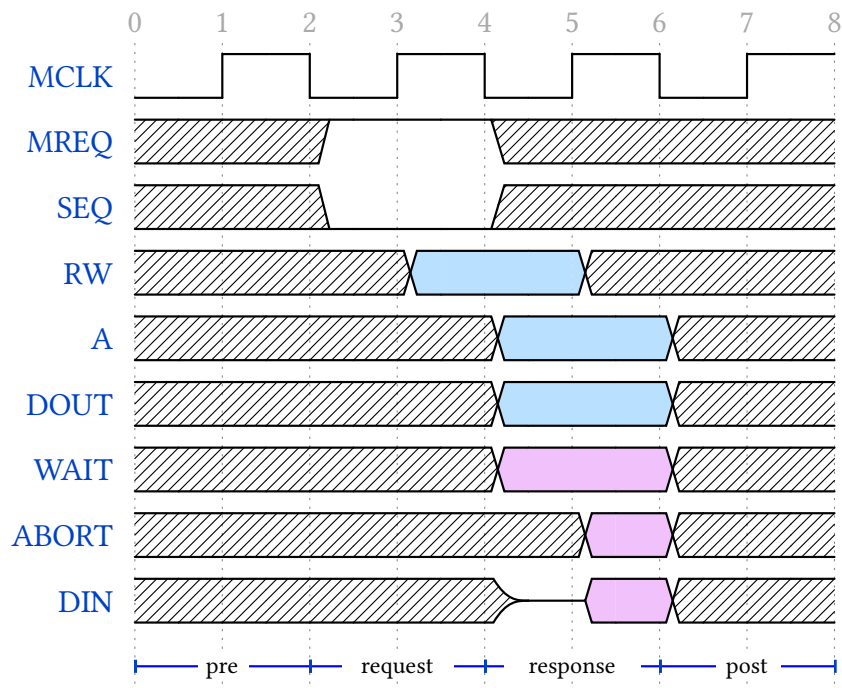


Figure 2: Nonsequential Memory Sequence timing diagram

- 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 (S-CYCLE)

RELATED PROCEDURES	
test_S_cycle	
gba_request_S_cycle	
memory_respond_S_cycle	
REQUEST SIGNALS	RESPONSE SIGNALS
MREQ , SEQ , RW , A , DOUT	WAIT , ABORT , DIN
DURATION: 2 CLOCK CYCLES	

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.

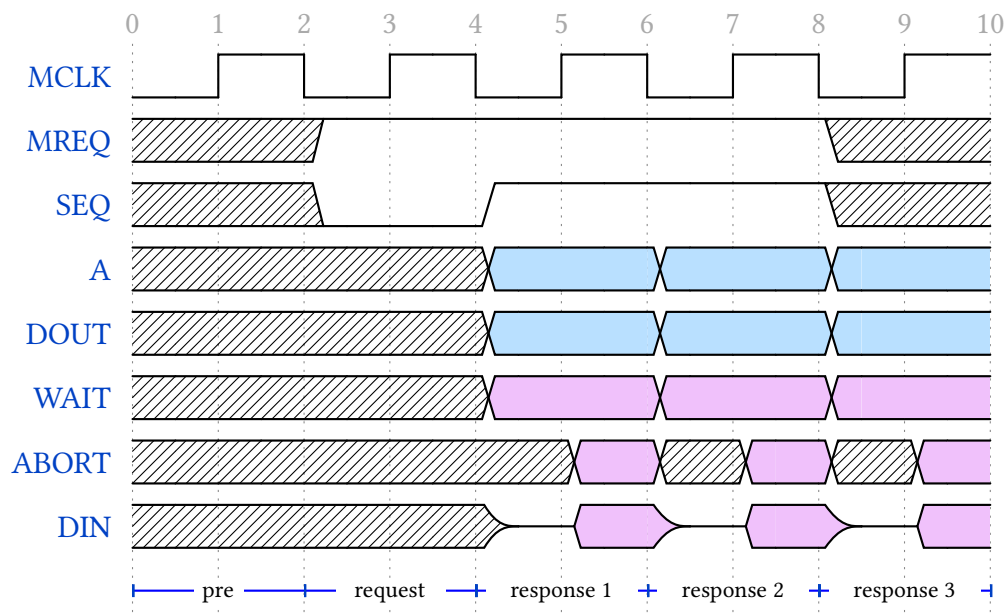


Figure 3: Sequential Memory Sequence timing diagram

- 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

gba\_initiate\_I\_cycle

### SIGNALS

MREQ , SEQ , A , DIN , DOUT

**DURATION:** 1 CLOCK CYCLE

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

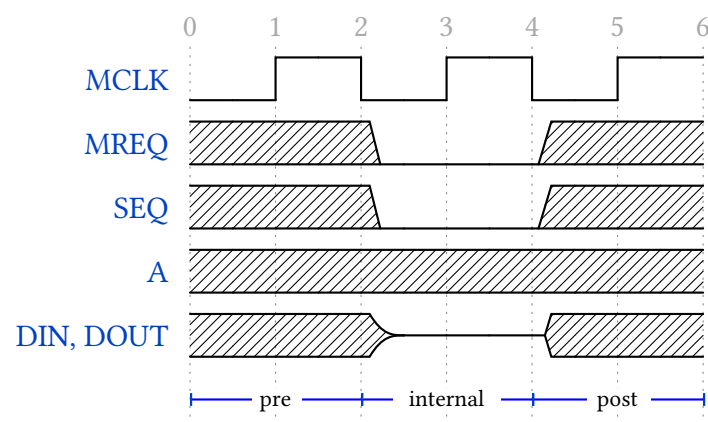


Figure 4: Internal Sequence timing diagram

## 5.5. MERGED INTERNAL-SEQUENTIAL SEQUENCE (MIS-CYCLE)

RELATED PROCEDURES	
test_MIS_cycle	
gba_request_MIS_cycle	
memory_respond_MIS_cycle	
REQUEST SIGNALS	RESPONSE SIGNALS
MREQ , SEQ , RW , A , DOUT	WAIT , ABORT , DIN
<b>DURATION:</b> 2 CLOCK CYCLES	

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

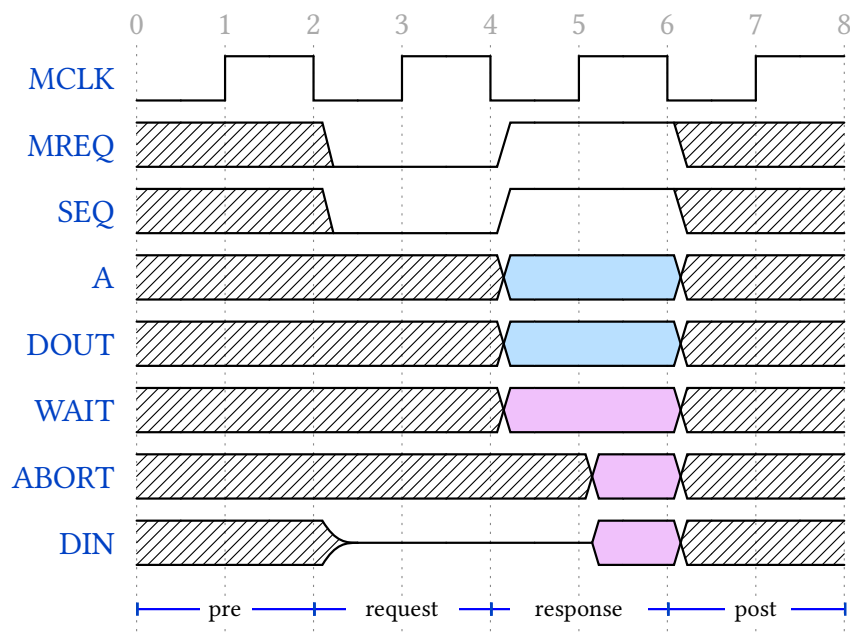


Figure 5: Merged Internal-Sequential Sequence timing diagram

## 5.6. DEPIPELINED ADDRESSING

### RELATED PROCEDURES

test\_depipedlined\_addressing

### REQUEST SIGNALS

MREQ , SEQ , RW , A , DOUT

### RESPONSE SIGNALS

WAIT , ABORT , DIN

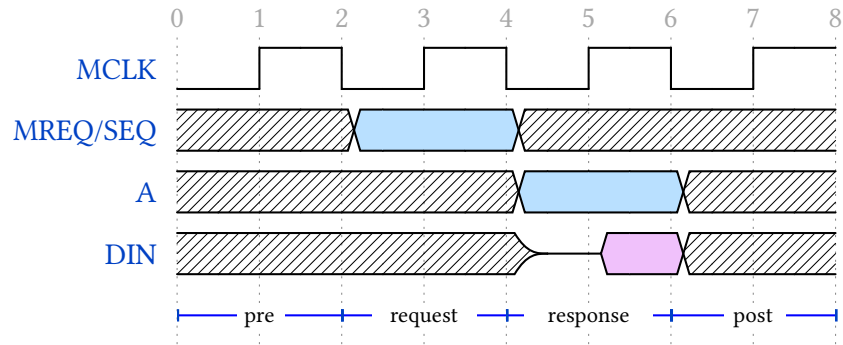


Figure 6: Depipelined Addressing timing diagram

## 5.7. DATA WRITE SEQUENCE (DW-CYCLE)

RELATED PROCEDURES	
test_DW_cycle	
gba_request_DW_cycle	
memory_respond_DW_cycle	
REQUEST SIGNALS	RESPONSE SIGNALS
MREQ , RW , A , DOUT	WAIT , ABORT
<b>DURATION:</b> 2 CLOCK CYCLES	

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

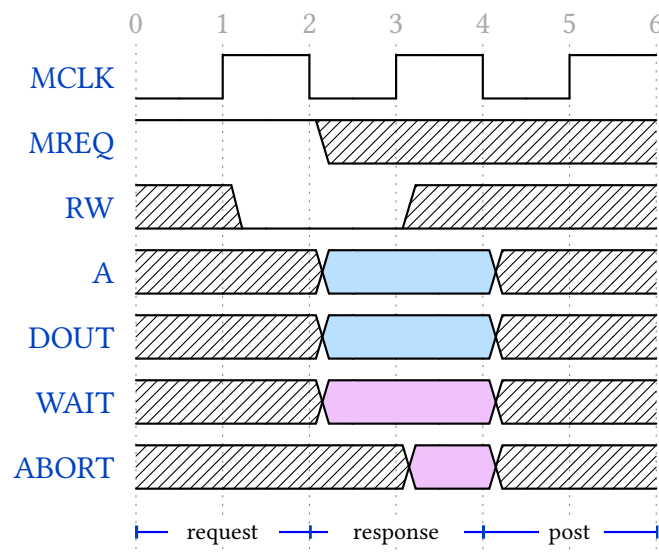


Figure 7: Data Write Sequence timing diagram

## 5.8. DATA READ SEQUENCE (DR-CYCLE)

RELATED PROCEDURES	
test_DR_cycle	
gba_request_DR_cycle	
memory_respond_DR_cycle	

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

<b>DURATION:</b> 2 CLOCK CYCLES
---------------------------------

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

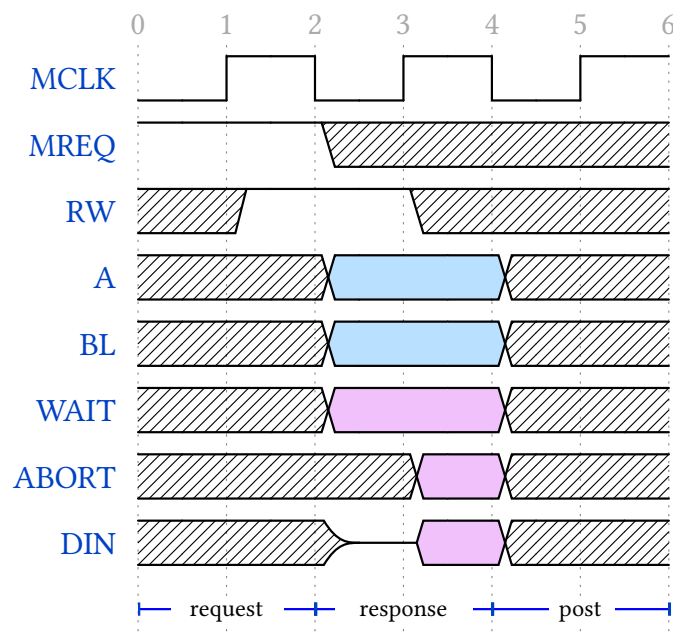


Figure 8: Data Read Sequence timing diagram



## 5.9. HALFWORD-WIDE MEMORY SEQUENCE

### RELATED PROCEDURES

test\_halfword\_memory\_sequence

gba\_request\_halfword\_memory\_sequence

memory\_respond\_halfword\_memory\_sequence

### REQUEST SIGNALS

MREQ , SEQ , RW , A , DOUT

### RESPONSE SIGNALS

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.

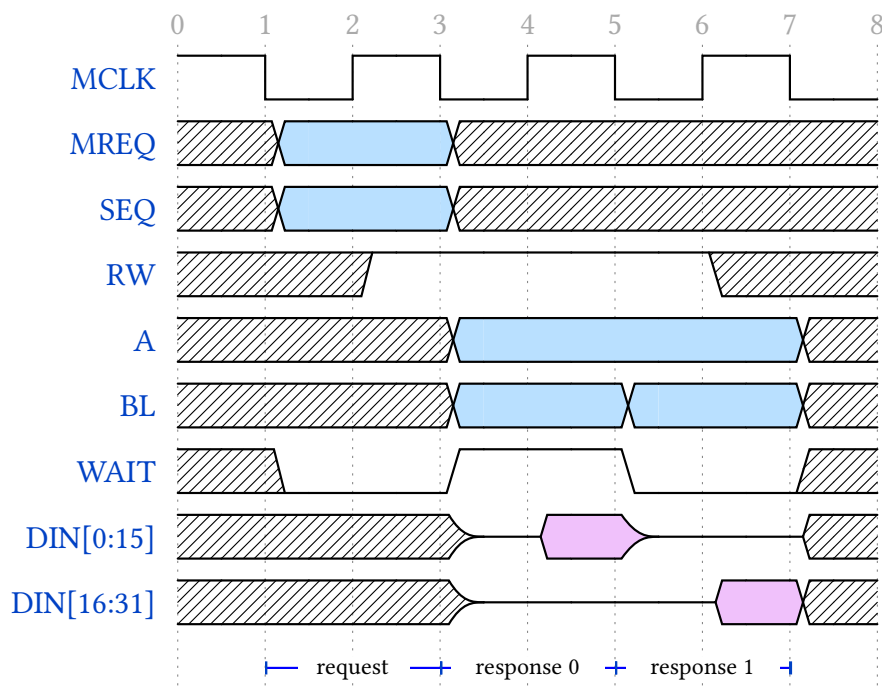


Figure 9: Halfword-Wide Read Memory Sequence timing diagram

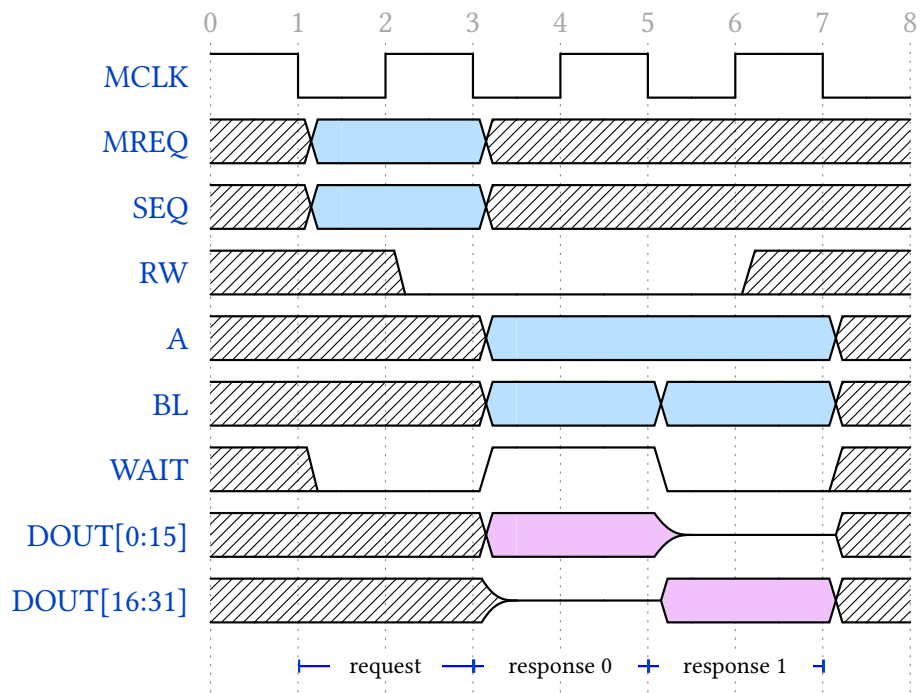


Figure 10: Halfword-Wide Write Memory Sequence timing diagram

## 5.10. BYTE-WIDE MEMORY SEQUENCE

RELATED PROCEDURES
test_byte_memory_sequence
gba_request_byte_memory_sequence
memory_respond_byte_memory_sequence

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.

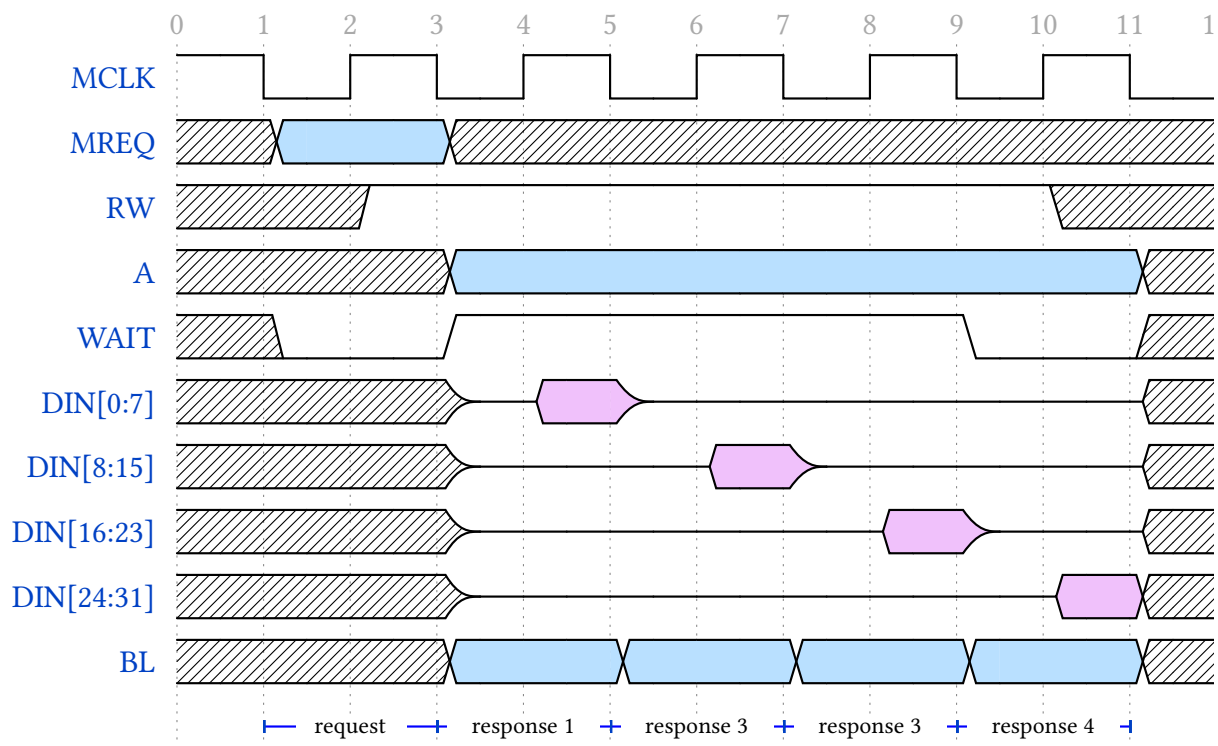


Figure 11: Byte-Wide Read Memory Sequence timing diagram

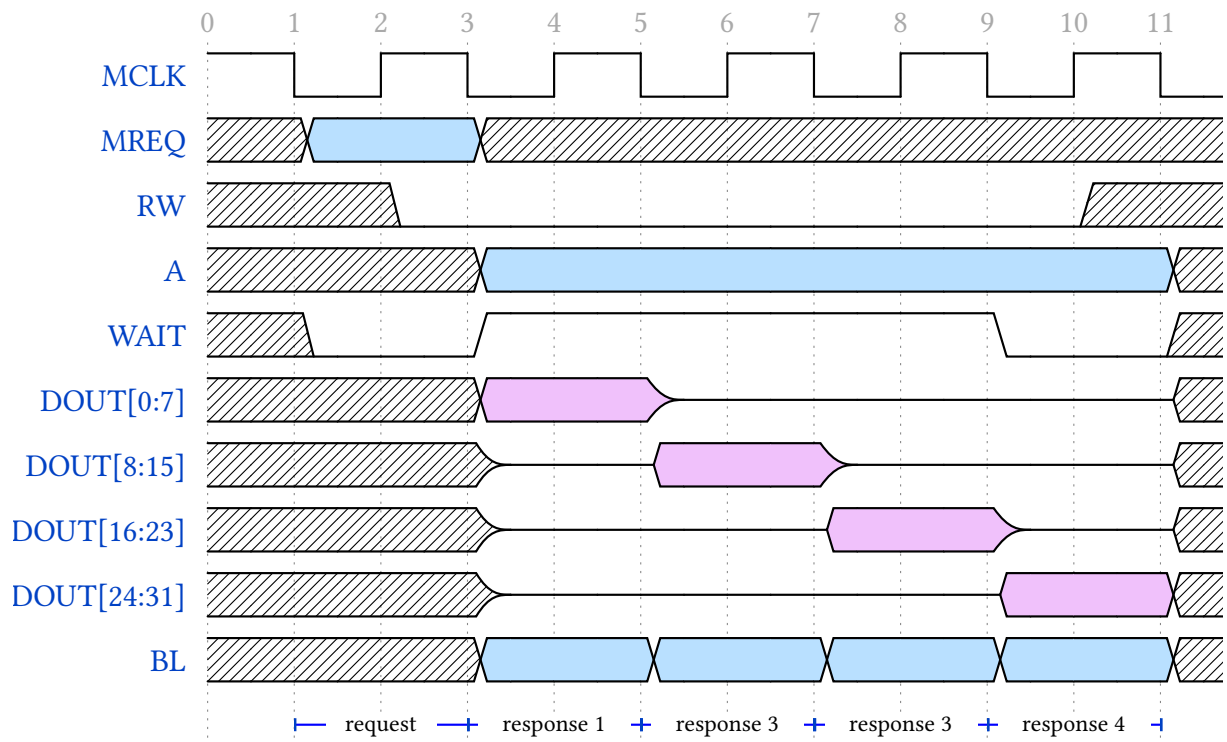


Figure 12: Byte-Wide Write Memory Sequence timing diagram

## 5.11. RESET SEQUENCE (RS-CYCLE)

### RELATED PROCEDURES

test\_RS\_cycle

gba\_initiate\_RS\_cycle

### REQUEST SIGNALS

MREQ , SEQ , RW , EXEC , OPC , A

### RESPONSE SIGNALS

WAIT , ABORT , DIN

**DURATION:** 5 CLOCK CYCLES

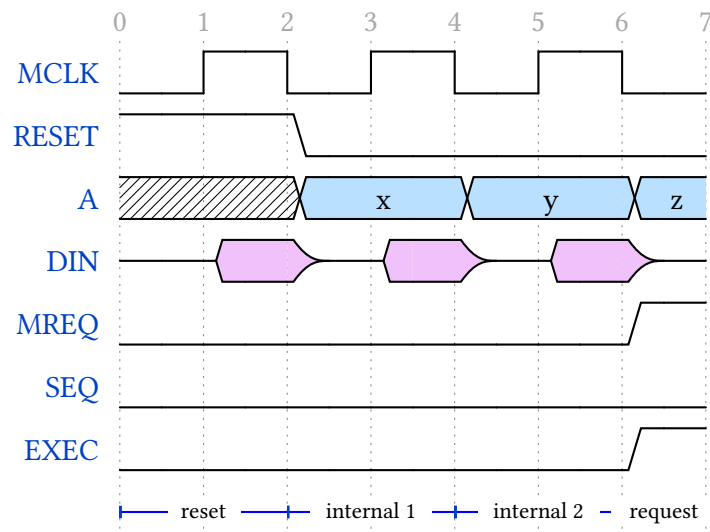


Figure 13: Reset Sequence timing diagram

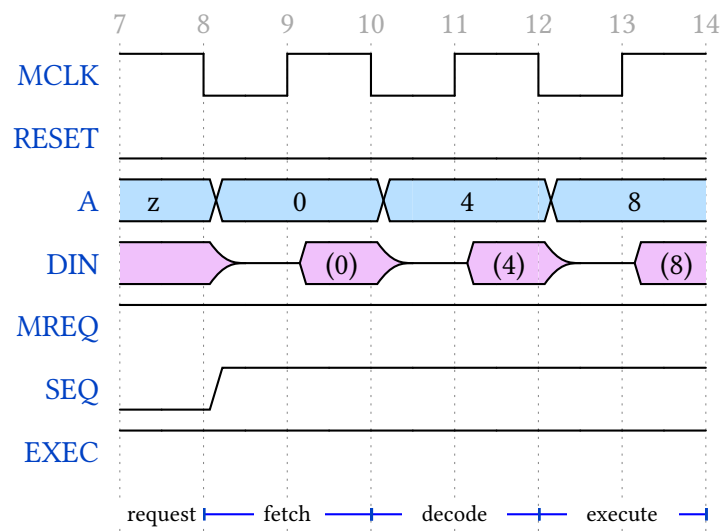


Figure 14: Reset Sequence timing diagram (continued)

## 5.12. GENERAL TIMING

### RELATED PROCEDURES

test\_general\_timing

### PHASE 1 SIGNALS

MREQ , SEQ , EXEC , EXEC , INSTRVALID ,  
A , BIGEND , ISYNC

### PHASE 2 SIGNALS

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.

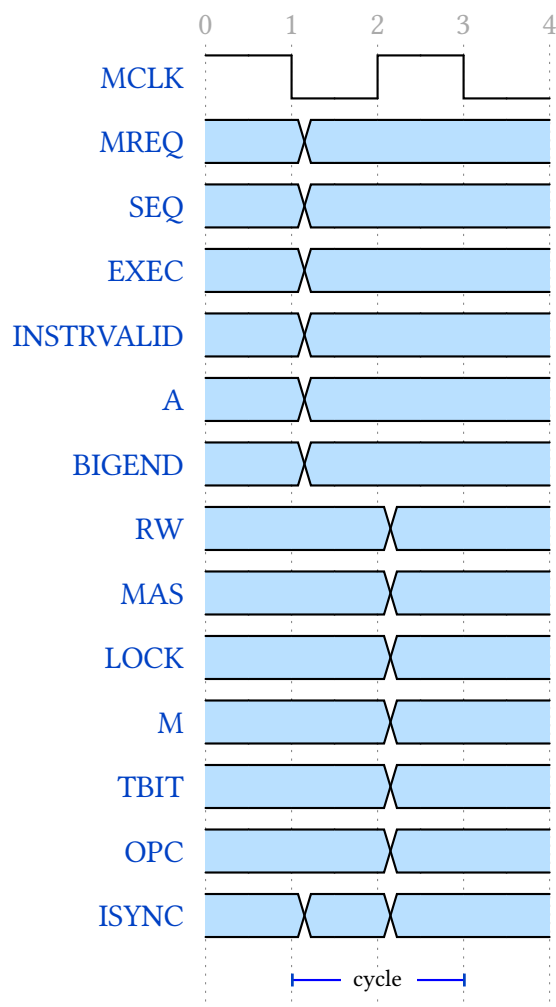


Figure 15: General timing diagram

## 5.13. ADDRESS BUS CONTROL

### RELATED PROCEDURES

test\_address\_bus\_control

### SIGNALS

ABE , A , RW , LOCK , OPC , MAS

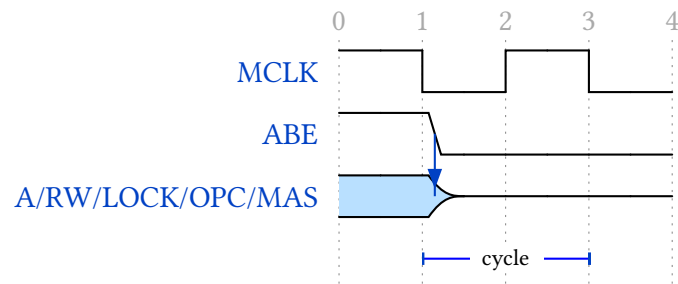


Figure 16: Address Bus Control timing diagram

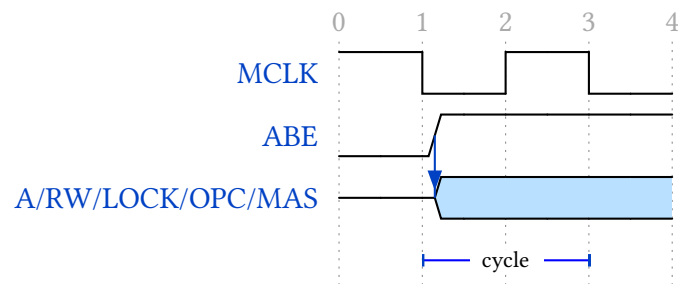


Figure 17: Address Bus Control timing diagram

- 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.

## 5.14. DATA BUS CONTROL

### RELATED PROCEDURES

test\_data\_bus\_control

### SIGNALS

DBE , DIN , DOUT

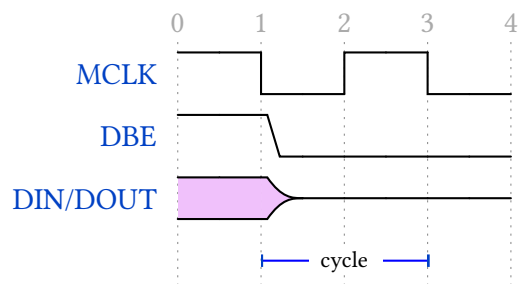


Figure 18: Data Bus Control timing diagram



## 5.15. EXCEPTION CONTROL

## RELATED PROCEDURES

test\_exception\_control

## SIGNALS

ABORT, FIQ, IRQ, RESET

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

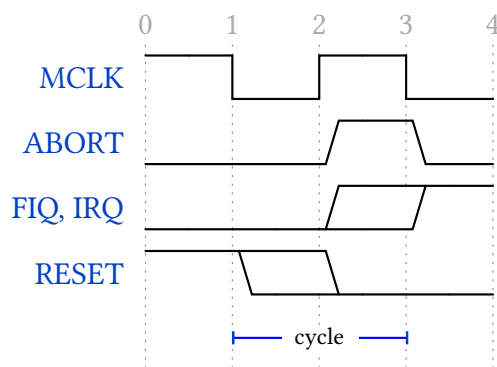


Figure 19: Exception Control timing diagram

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.

## 5.16. ADDRESS PIPELINE CONTROL

### RELATED PROCEDURES

test\_address\_pipeline\_control

### SIGNALS

APE , A , RW , LOCK , OPC , MAS

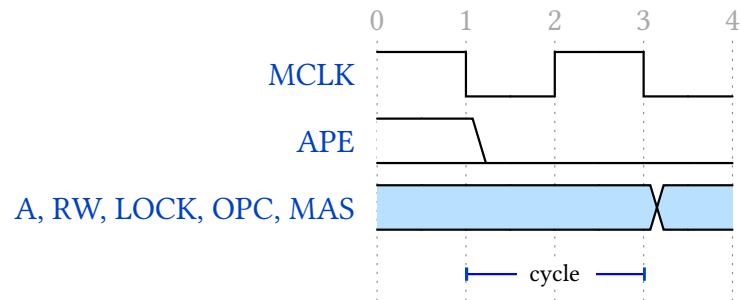


Figure 20: Address Pipeline Control timing diagram

## 5.17. GENERAL INSTRUCTION CYCLE

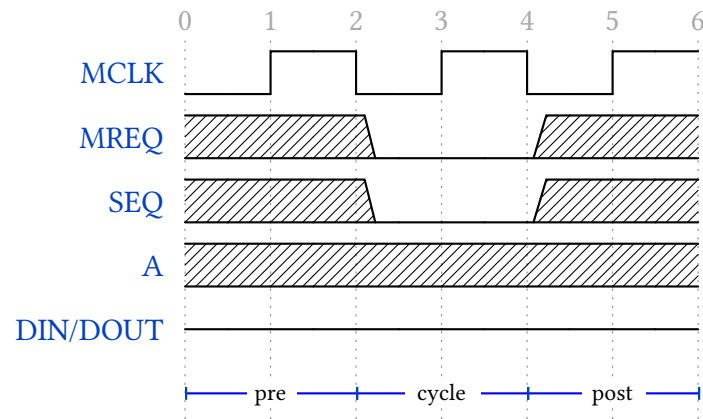


Figure 21: General Instruction Cycle timing diagram

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*).

## 5.18. BRANCH AND BRANCH WITH LINK INSTRUCTION CYCLE (BABLI-CYCLE)

### RELATED INSTRUCTIONS

B, BL

### RELATED PROCEDURES

test\_BABLI\_cycle

gba\_request\_BABLI\_cycle

memory\_respond\_BABLI\_cycle

**DURATION:** 3 CLOCK CYCLES

	PARAMETERS
pc	Program counter, before executing the instruction.
L	Instruction length, 4 for ARM state, 2 for Thumb state.
alu	The instruction operand—ie, the address to jump to.
i	MAS, 2 for ARM state, 1 for Thumb state.

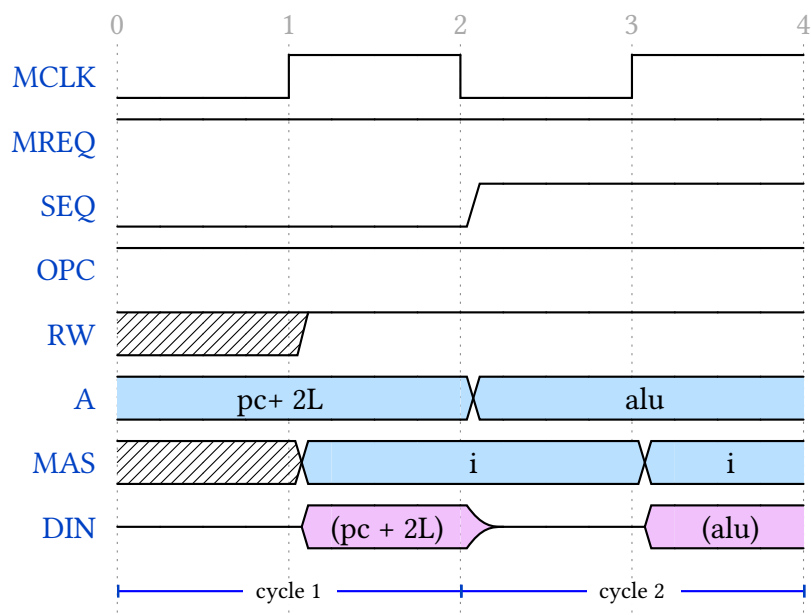


Figure 22: Branch and Branch with Link Instruction Cycle

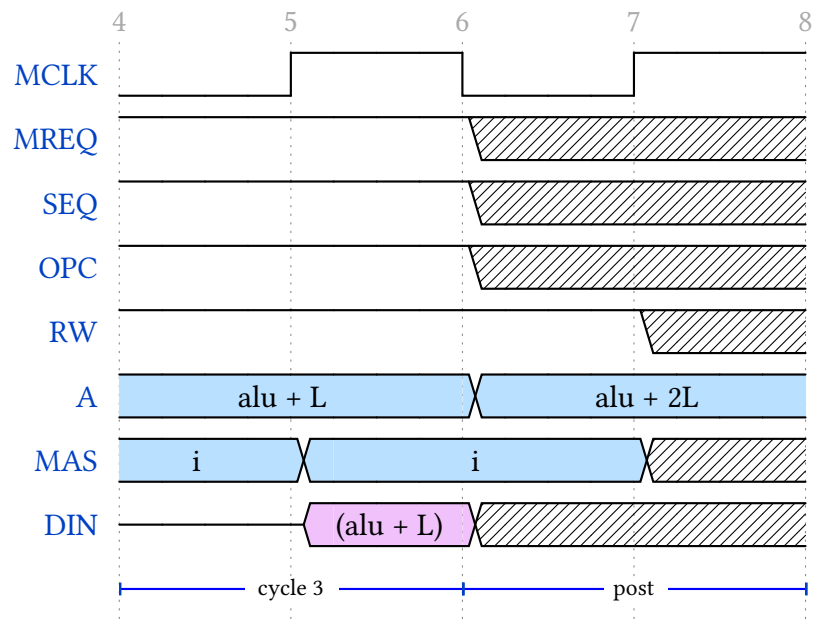


Figure 23: Branch and Branch with Link Instruction Cycle (continued)

# 5.19. THUMB BRANCH WITH LINK INSTRUCTION CYCLE (TBLI-CYCLE)

RELATED INSTRUCTIONS

RELATED PROCEDURES
test_TBLI_cycle
gba_request_TBLI_cycle
memory_respond_TBLI_cycle

DURATION: 4 CLOCK CYCLES
--------------------------

	PARAMETERS
pc	Program counter, before executing the instruction.
alu	The instruction operand—ie, the address to jump to.

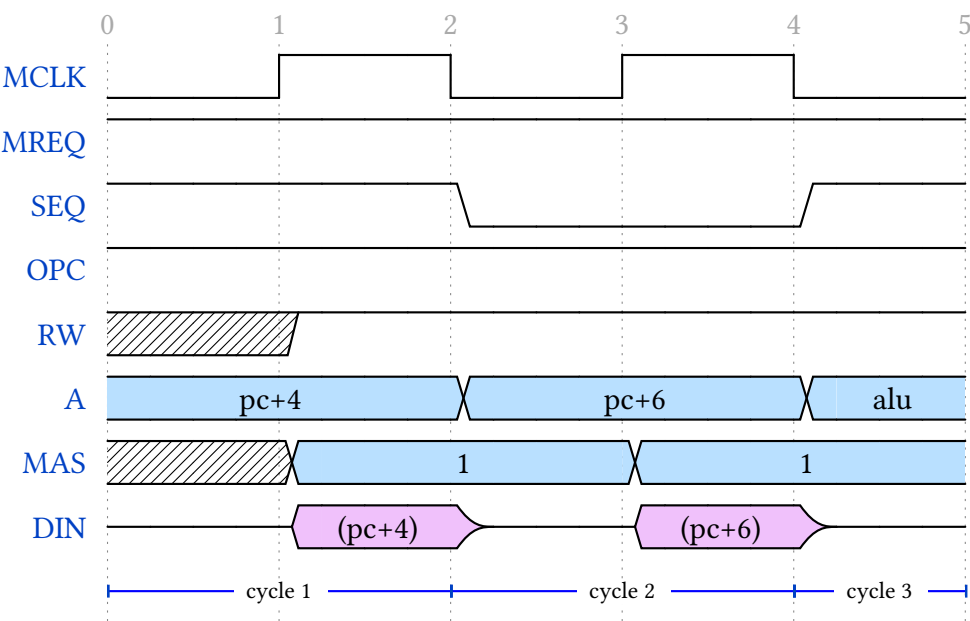


Figure 24: Thumb Branch with Link Instruction Cycle

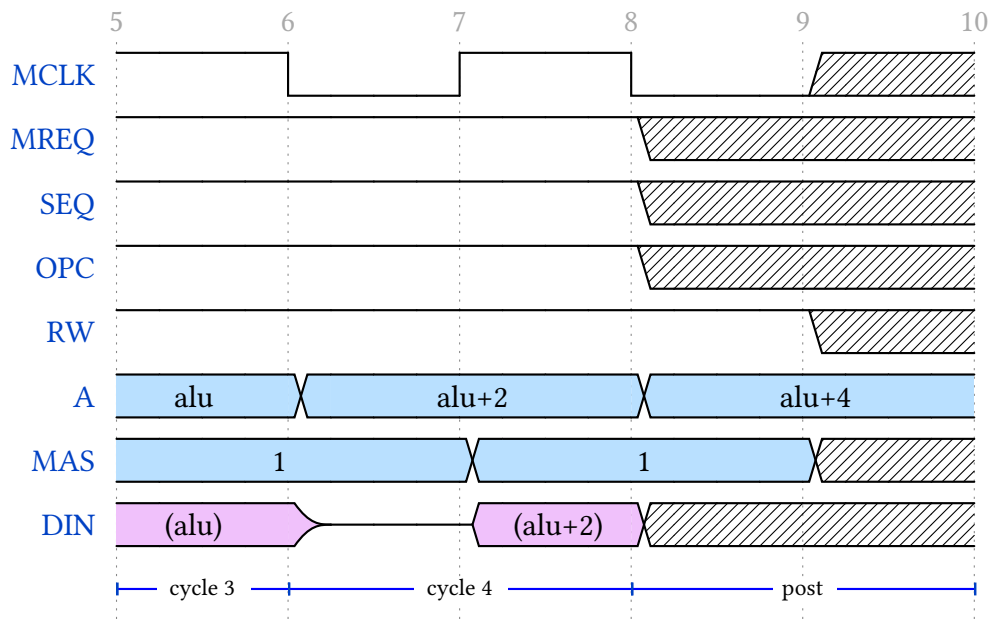


Figure 25: Thumb Branch with Link Instruction Cycle (continued)

## 5.20. BRANCH AND EXCHANGE INSTRUCTION CYCLE (BAEI-CYCLE)

### RELATED INSTRUCTIONS

BX

### RELATED PROCEDURES

test\_BAEI\_cycle

gba\_request\_BAEI\_cycle

memory\_respond\_BAEI\_cycle

**DURATION:** 3 CLOCK CYCLES

	PARAMETERS
pc	Program counter, before executing the instruction.
alu	The instruction operand—ie, the address to jump to.
I	MAS before executing the instruction.
i	MAS after executing the instruction.
W	Instruction width before executing the instruction.
w	Instruction width after executing the instruction.
T	TBIT before executing the instruction.
t	TBIT after executing the instruction.

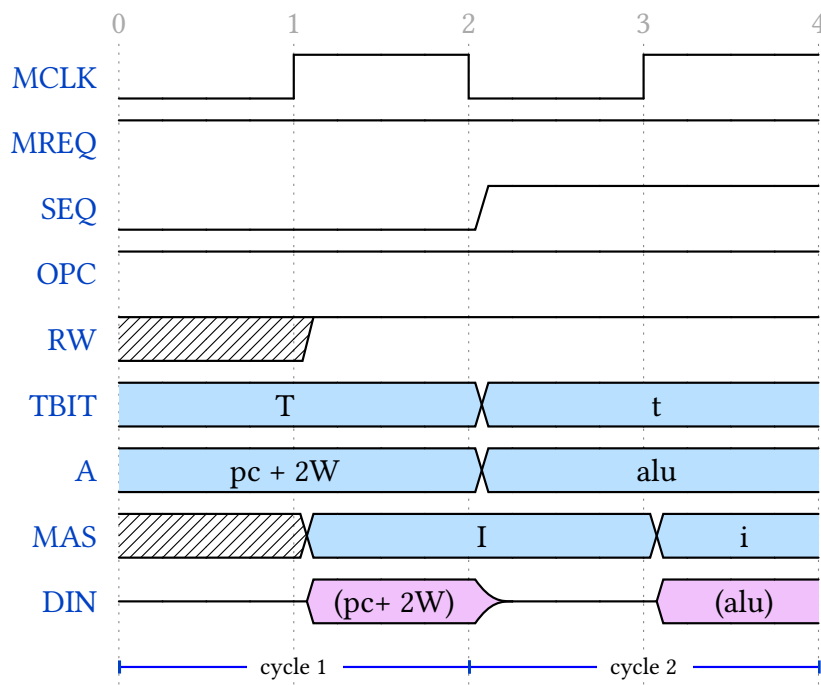


Figure 26: Branch and Exchange Instruction Cycle



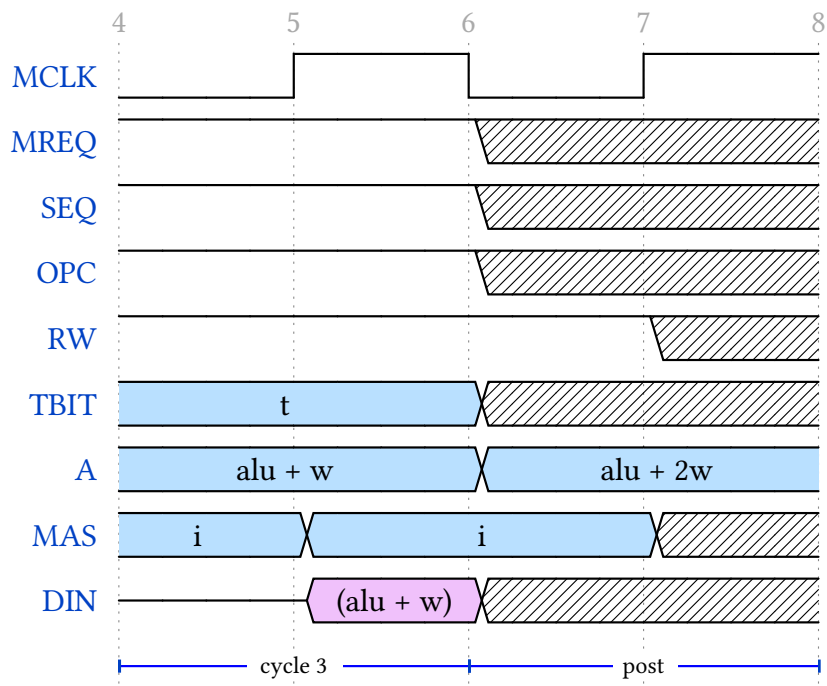


Figure 27: Branch and Exchange Instruction Cycle (continued)

## 5.21. DATA PROCESSING INSTRUCTION CYCLE (DPI-CYCLE)

### RELATED INSTRUCTIONS

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

### RELATED PROCEDURES

test\_DPI\_cycle

gba\_request\_DPI\_cycle

memory\_respond\_DPI\_cycle

**DURATION:** 1 TO 4 CLOCK CYCLES

	PARAMETERS
pc	Program counter, before executing the instruction.
L	Instruction length, 4 for ARM state, 2 for Thumb state.
alu	The instruction operand—ie, the address of the shifter operand.
i	MAS, 2 for ARM state, 1 for Thumb state.

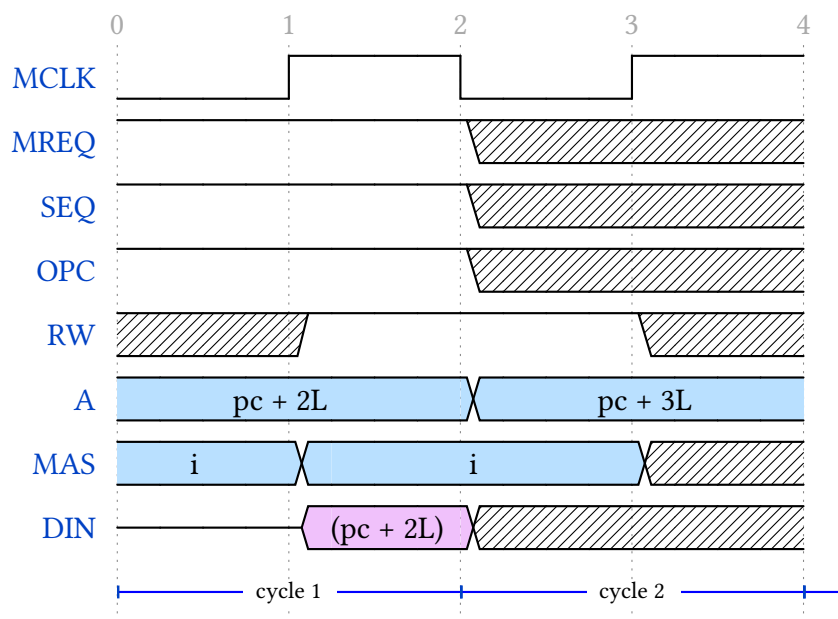


Figure 28: Data Processing Instruction Cycle (normal)

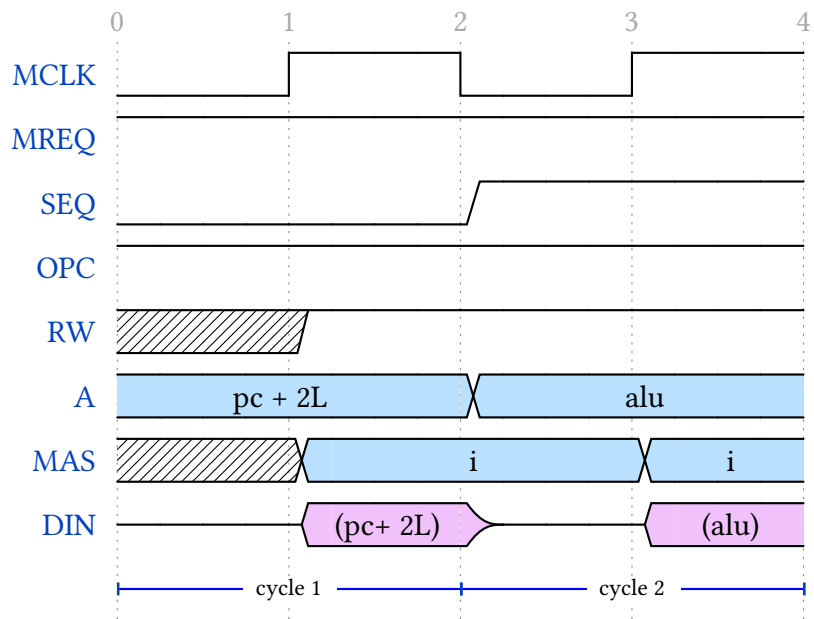


Figure 29: Data Processing Instruction Cycle (dest=pc)

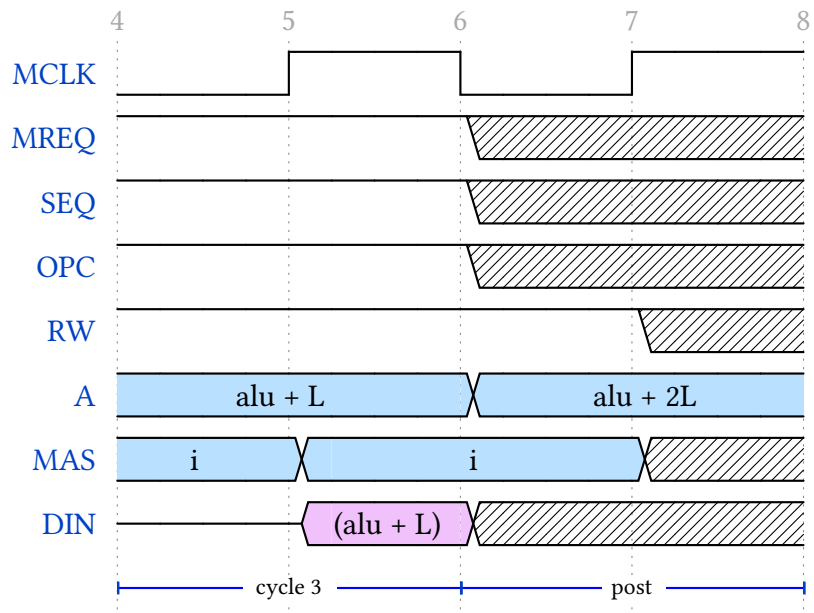


Figure 30: Data Processing Instruction Cycle (dest=pc) (continued)

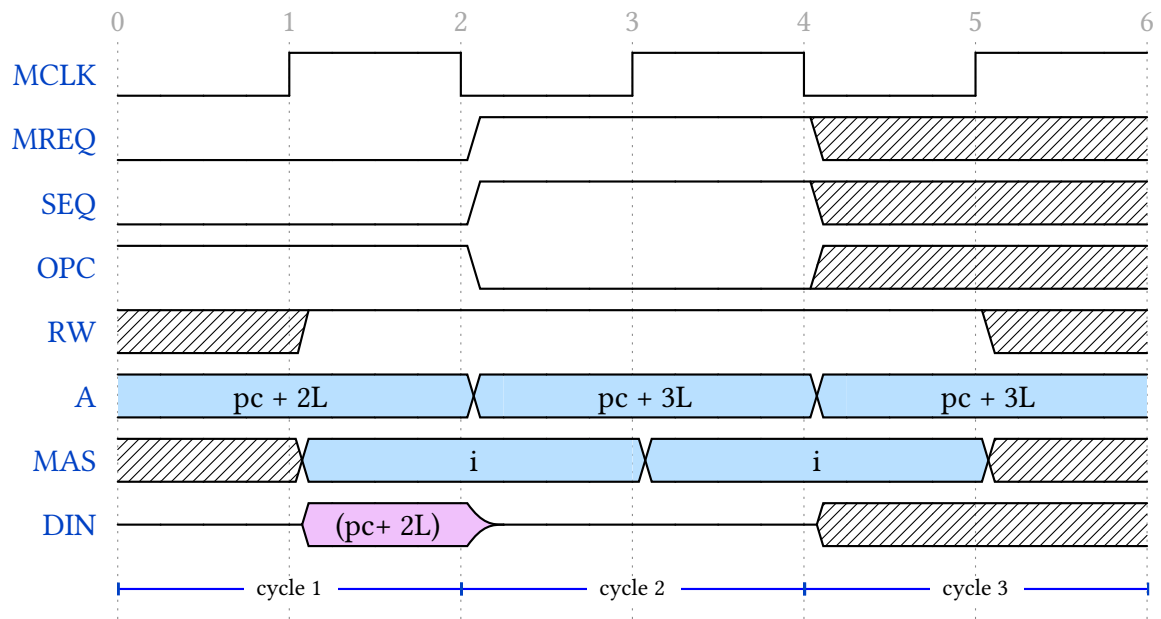


Figure 31: Data Processing Instruction Cycle (shift(RS))

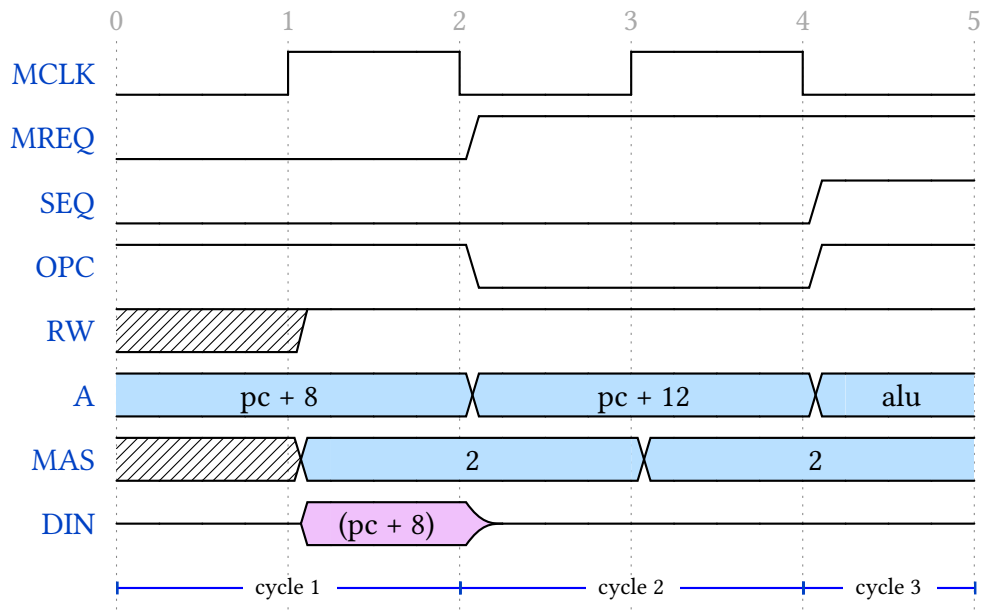


Figure 32: Data Processing Instruction Cycle (shift(Rs) dest=pc)

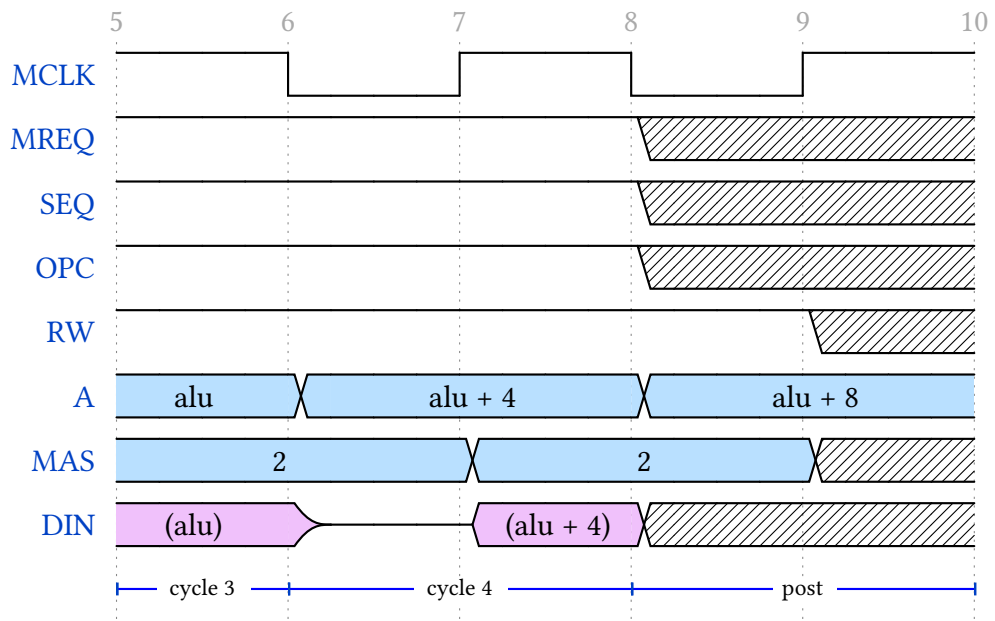


Figure 33: Data Processing Instruction Cycle (shift(Rs) dest=pc) (continued)

## 5.22. MULTIPLY AND MULTIPLY ACCUMULATE INSTRUCTION CYCLE (MAMAI-Cycle)

### RELATED INSTRUCTIONS

MLA , MUL , SMLAL , SMULL , UMLAL , UMULL

### RELATED PROCEDURES

test\_MAMAI\_cycle

gba\_request\_MAMAI\_cycle

memory\_respond\_MAMAI\_cycle

**DURATION:** 3 TO 4 CLOCK CYCLES

	PARAMETERS
pc	Program counter, before executing the instruction.
L	Instruction length, 4 for ARM state, 2 for Thumb state.
i	MAS, 2 for ARM state, 1 for Thumb state.

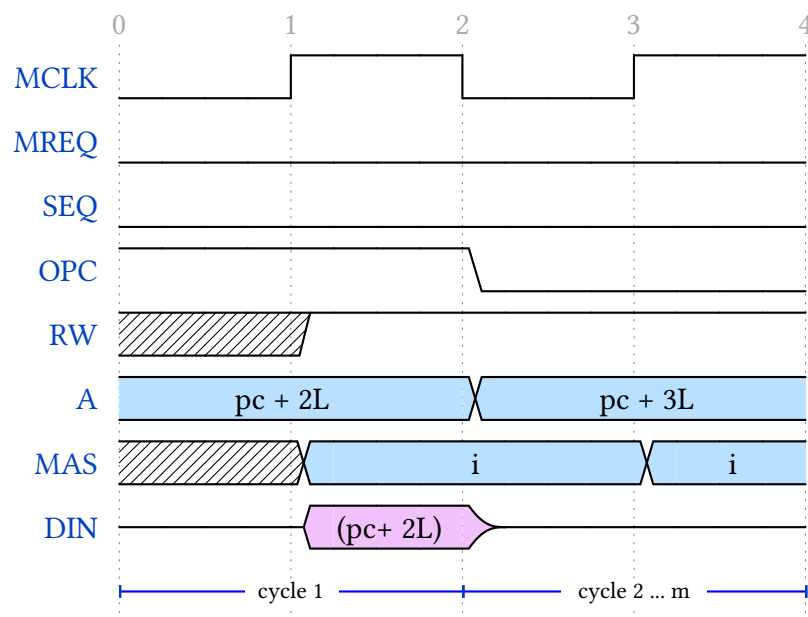


Figure 34: Multiply Instruction Cycle

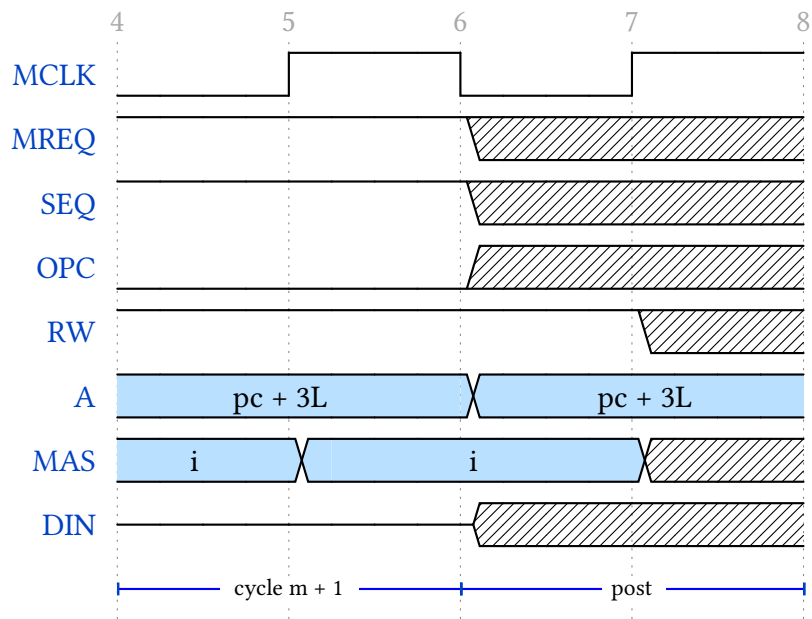


Figure 35: Multiply Instruction Cycle (continued)

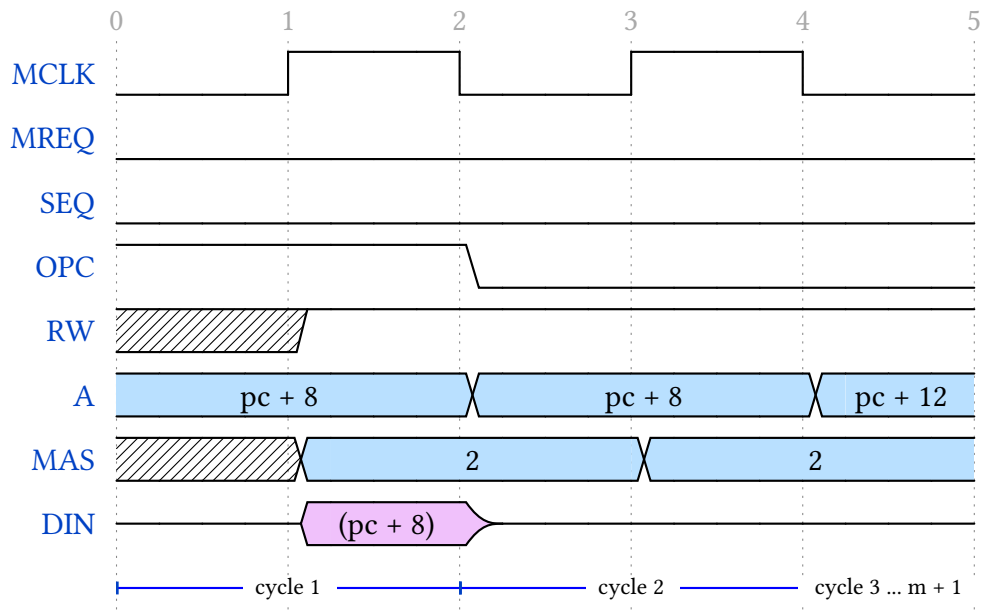


Figure 36: Multiply Accumulate Instruction Cycle

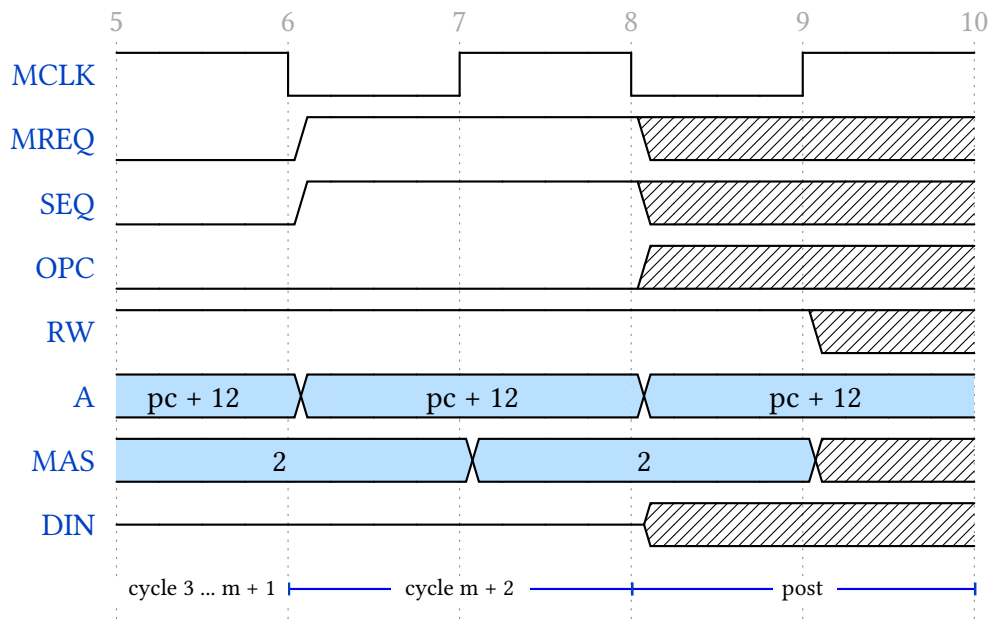


Figure 37: Multiply Accumulate Instruction Cycle (continued)

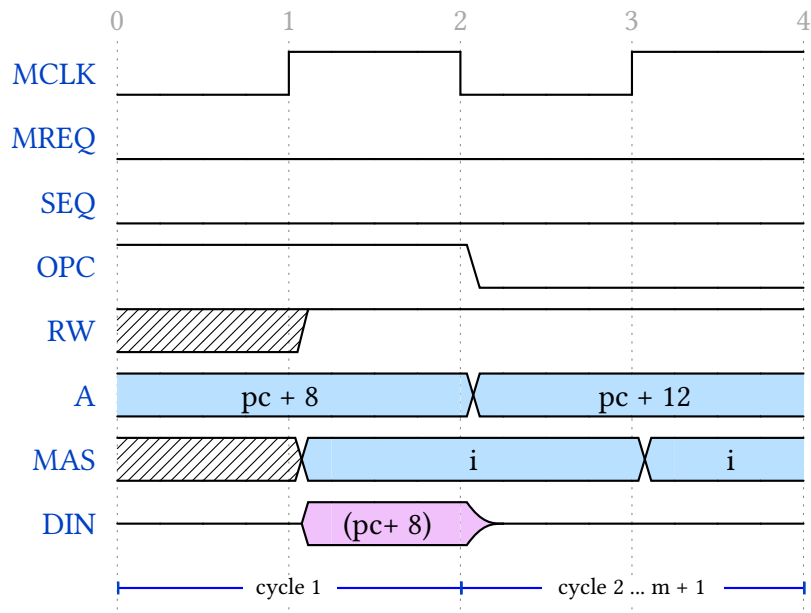


Figure 38: Multiply Long Instruction Cycle



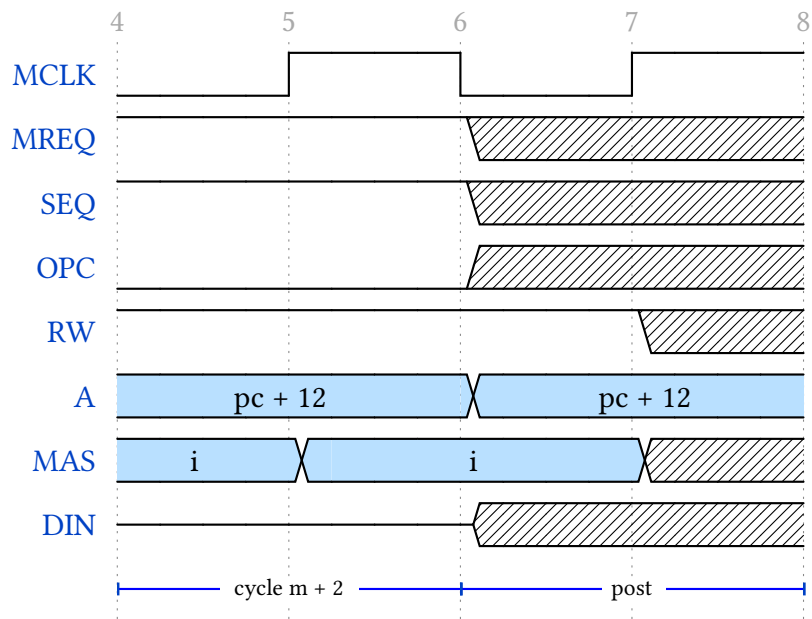


Figure 39: Multiply Long Instruction Cycle (continued)

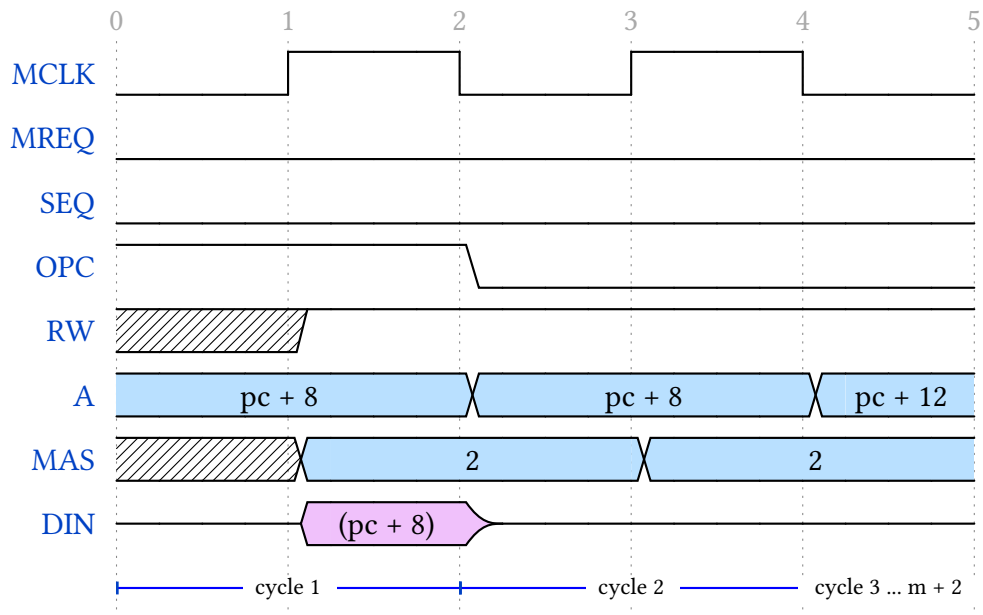


Figure 40: Multiply Accumulate Long Instruction Cycle

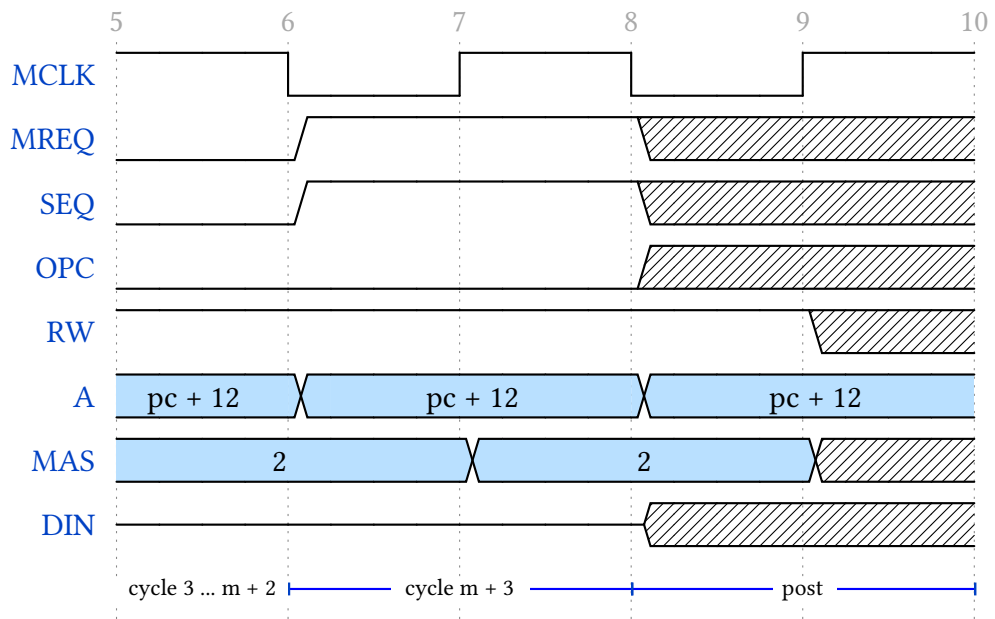


Figure 41: Multiply Accumulate Long Instruction Cycle (continued)

## 5.23. LOAD REGISTER INSTRUCTION CYCLE (LRI-CYCLE)

### RELATED INSTRUCTIONS

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

### RELATED PROCEDURES

test\_LRI\_cycle

gba\_request\_LRI\_cycle

memory\_respond\_LRI\_cycle

**DURATION:** 3 TO 5 CLOCK CYCLES

### PARAMETERS

pc	Program counter, before executing the instruction.
L	Instruction length, 4 for ARM state, 2 for Thumb state.
alu	The instruction operand—ie, the first source address.
i	MAS, 2 for ARM state, 1 for Thumb state.
s	MAS variable ( BYTE , HALFWORD , or WORD ).

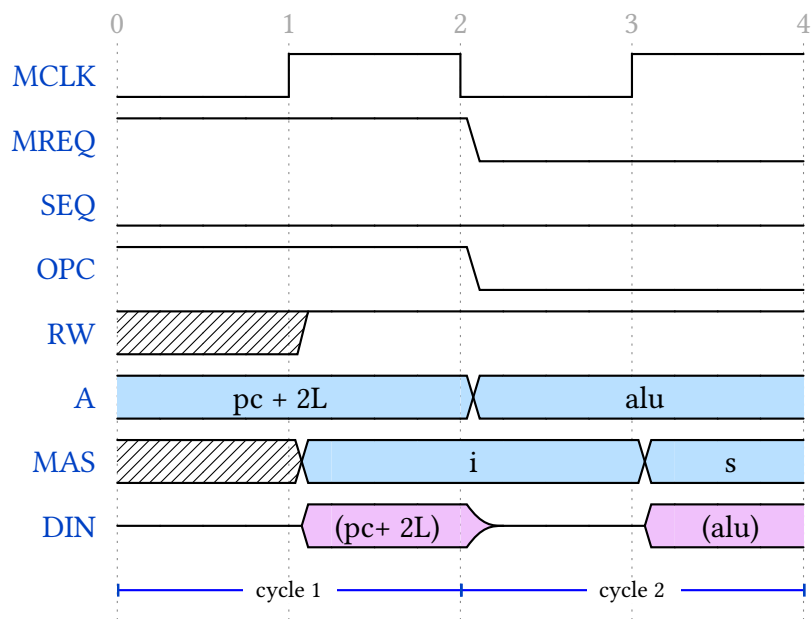


Figure 42: Load Register Instruction Cycle (normal)

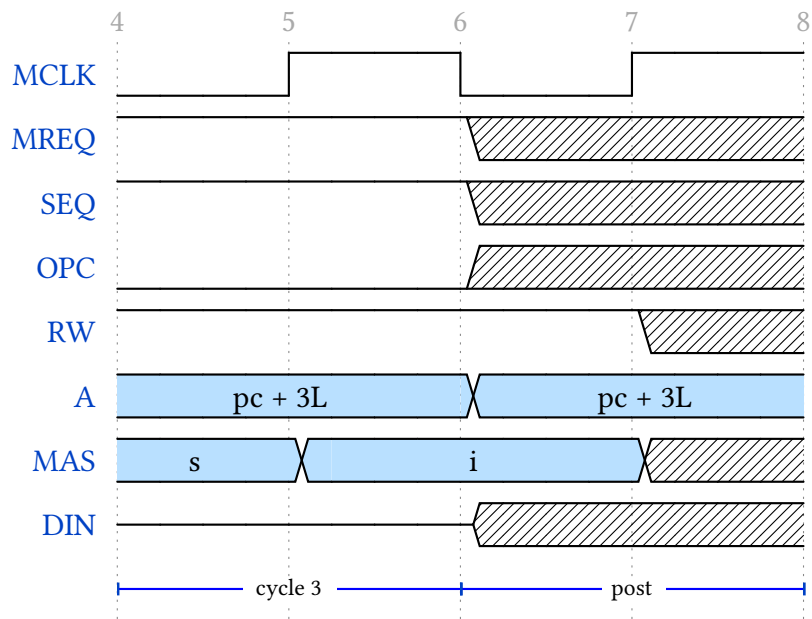


Figure 43: Load Register Instruction Cycle (normal) (continued)

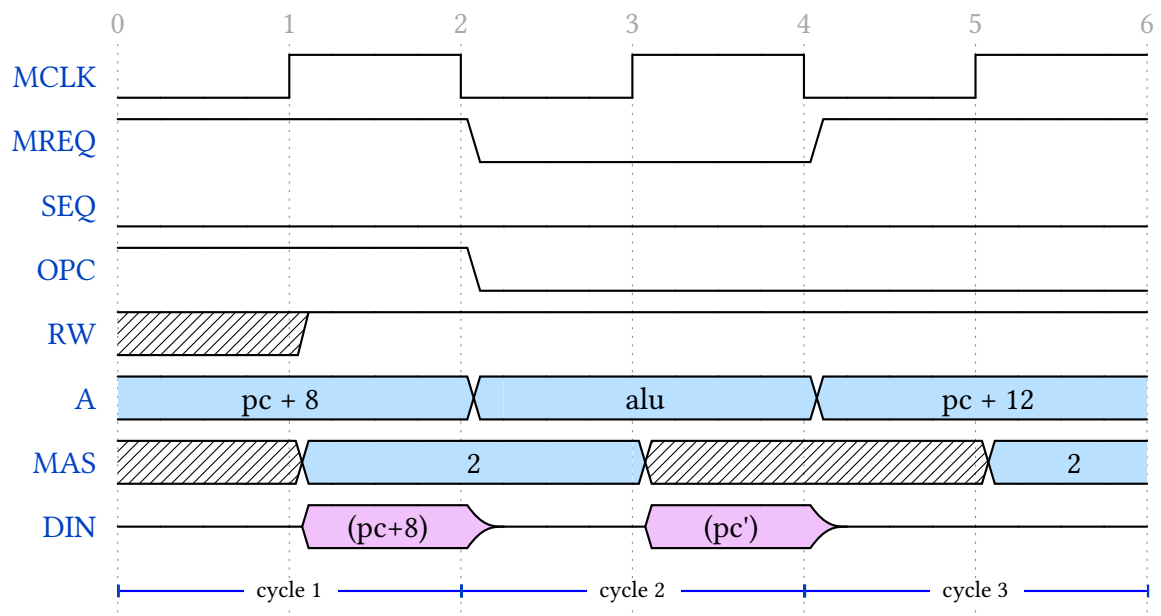


Figure 44: Load Register Instruction Cycle (dest=pc)

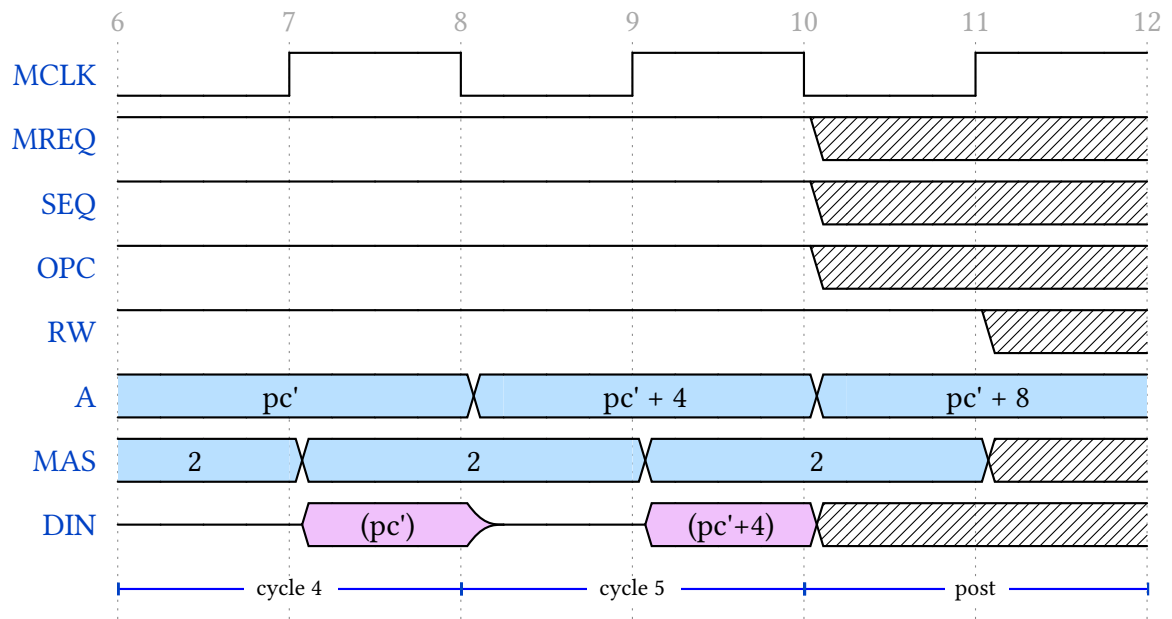


Figure 45: Load Register Instruction Cycle (dest=pc) (continued)

## 5.24. STORE REGISTER INSTRUCTION CYCLE (SRI-CYCLE)

### RELATED INSTRUCTIONS

STR , STRB , STRBT , STRH , STRT

### RELATED PROCEDURES

test\_SRI\_cycle

gba\_request\_SRI\_cycle

memory\_respond\_SRI\_cycle

**DURATION:** 2 CLOCK CYCLES

### PARAMETERS

pc	Program counter, before executing the instruction.
L	Instruction length, 4 for ARM state, 2 for Thumb state.
alu	The instruction operand—ie, the first target address.
i	MAS, 2 for ARM state, 1 for Thumb state.
s	MAS variable ( BYTE , HALFWORD , or WORD ).

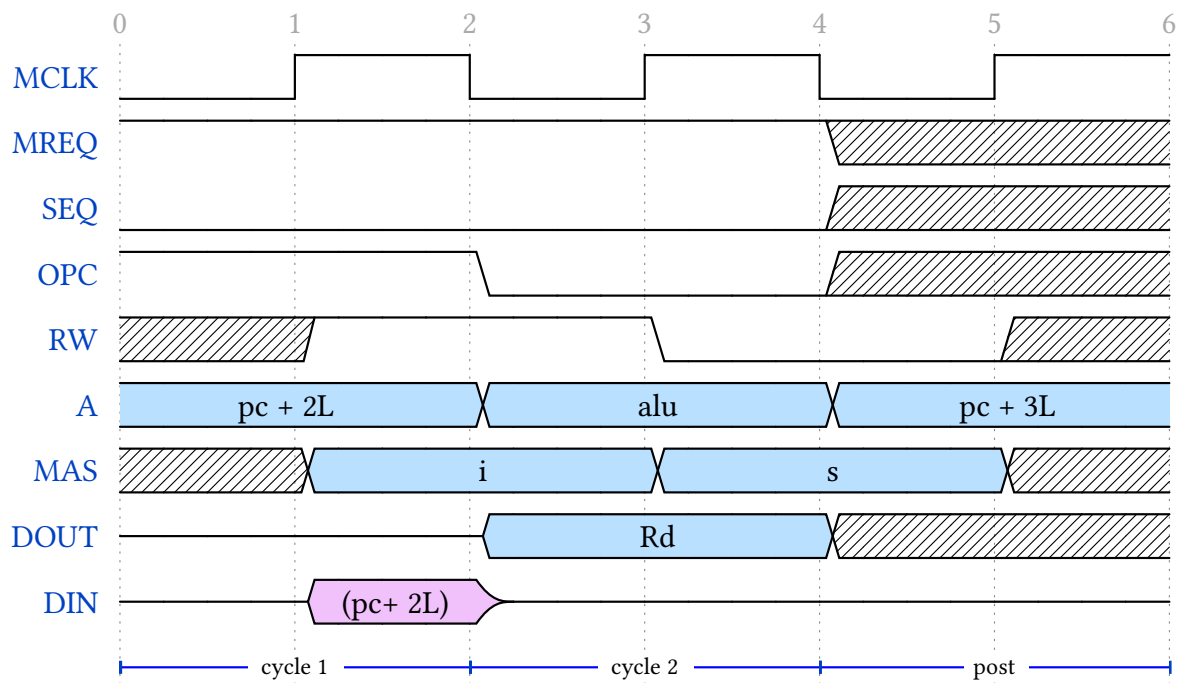


Figure 46: Store Register Instruction Cycle

# 5.25. LOAD MULTIPLE REGISTER INSTRUCTION CYCLE (LMRI-CYCLE)

RELATED INSTRUCTIONS
LDM

RELATED PROCEDURES
test_LMRI_cycle
gba_request_LMRI_cycle
memory_respond_LMRI_cycle

DURATION: 3 TO 6 CLOCK CYCLES
-------------------------------

	PARAMETERS
pc	Program counter, before executing the instruction.
L	Instruction length, 4 for ARM state, 2 for Thumb state.
alu	The instruction operand—ie, the first source address.
i	MAS, 2 for ARM state, 1 for Thumb state.
s	MAS variable ( BYTE , HALFWORD , or WORD ).

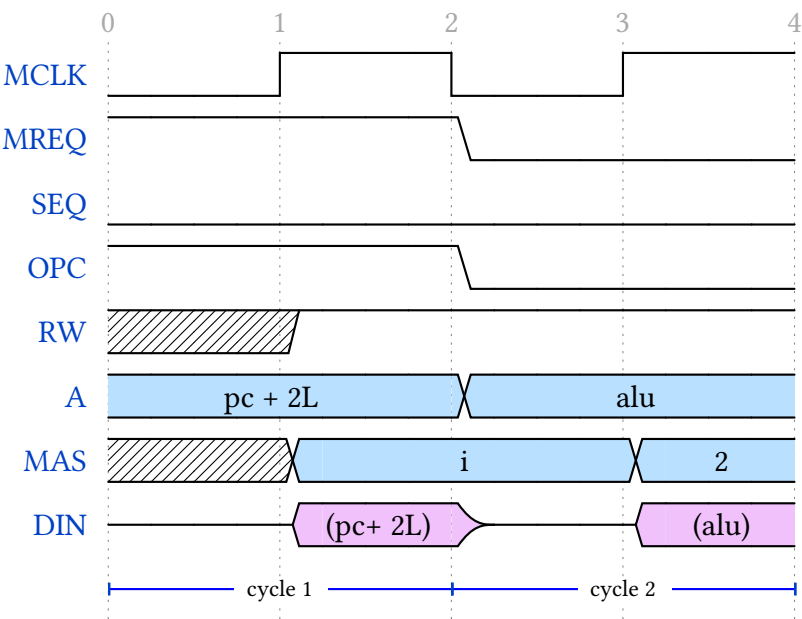


Figure 47: Load Multiple Register Instruction Cycle (single register)

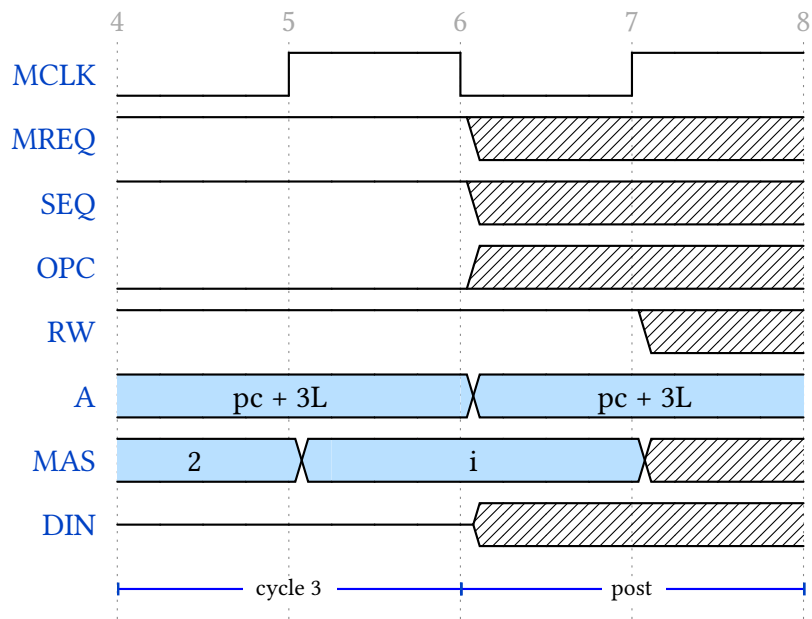


Figure 48: Load Multiple Register Instruction Cycle (single register) (continued)

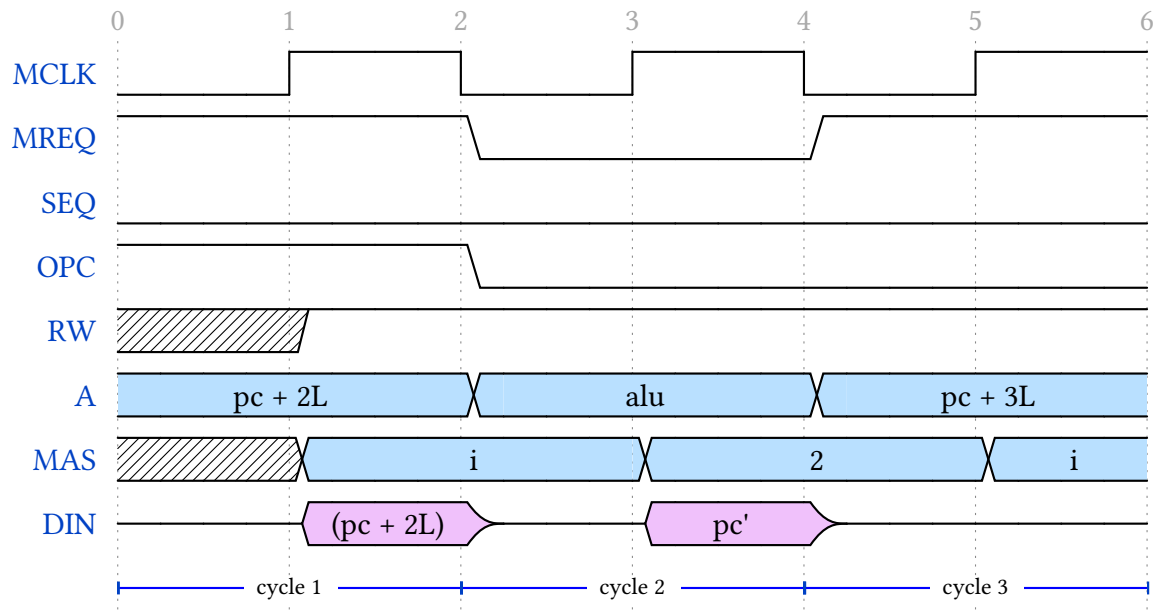


Figure 49: Load Multiple Register Instruction Cycle (single register dest=pc)



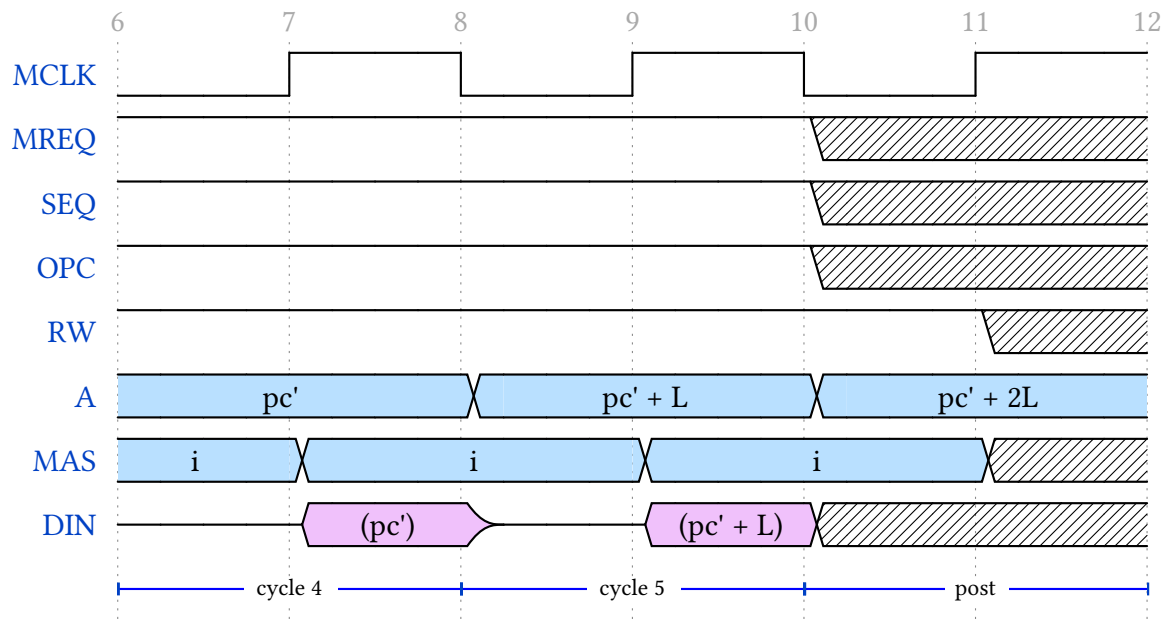


Figure 50: Load Multiple Register Instruction Cycle (single register dest=pc) (continued)

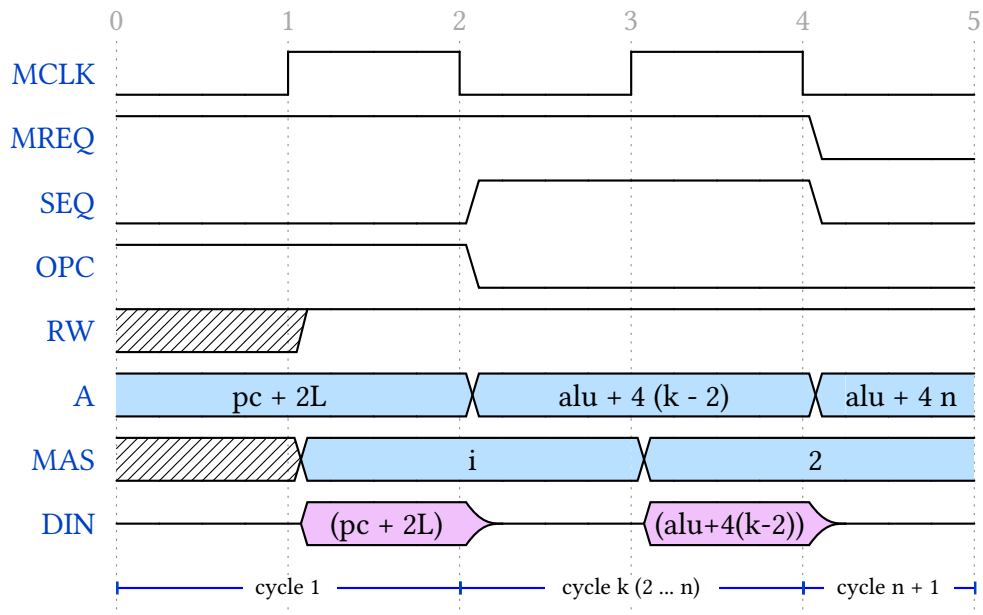


Figure 51: Load Multiple Register Instruction Cycle (n registers)

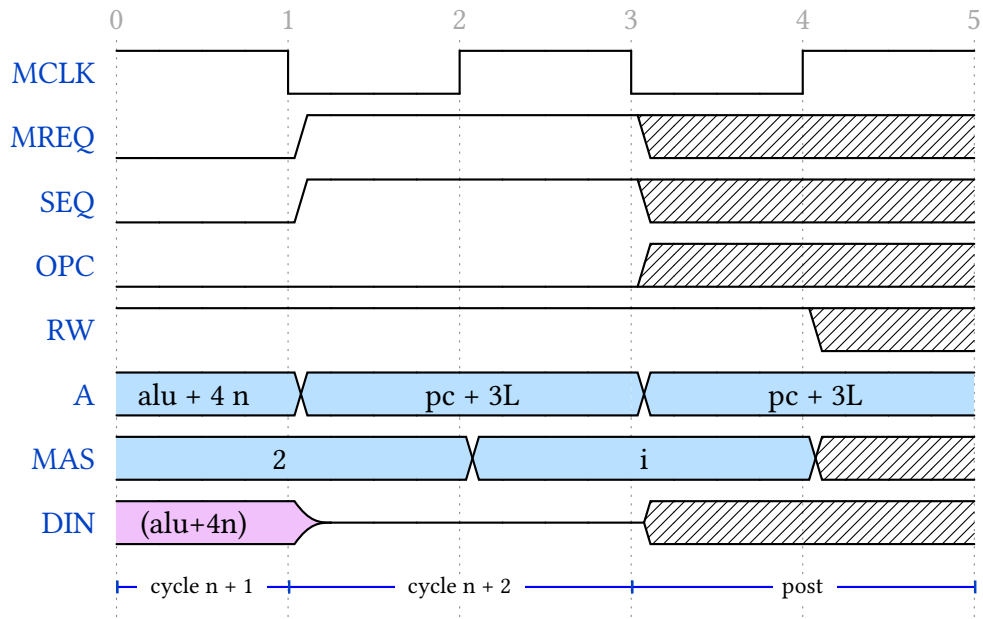


Figure 52: Load Multiple Register Instruction Cycle (n registers) (continued)

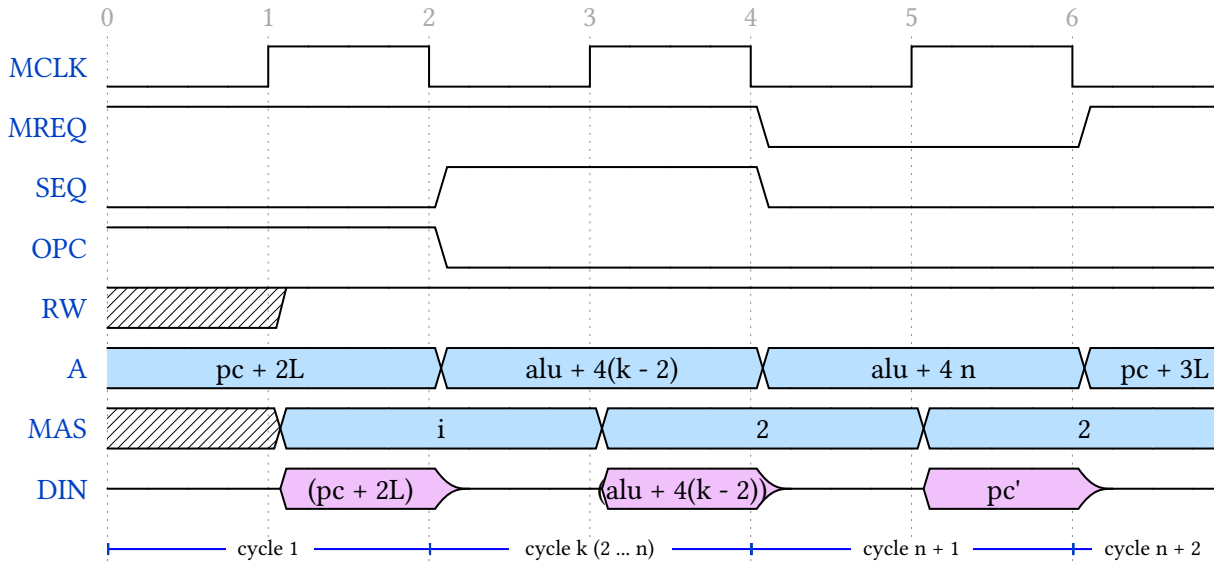


Figure 53: Load Multiple Register Instruction Cycle (n registers including pc)

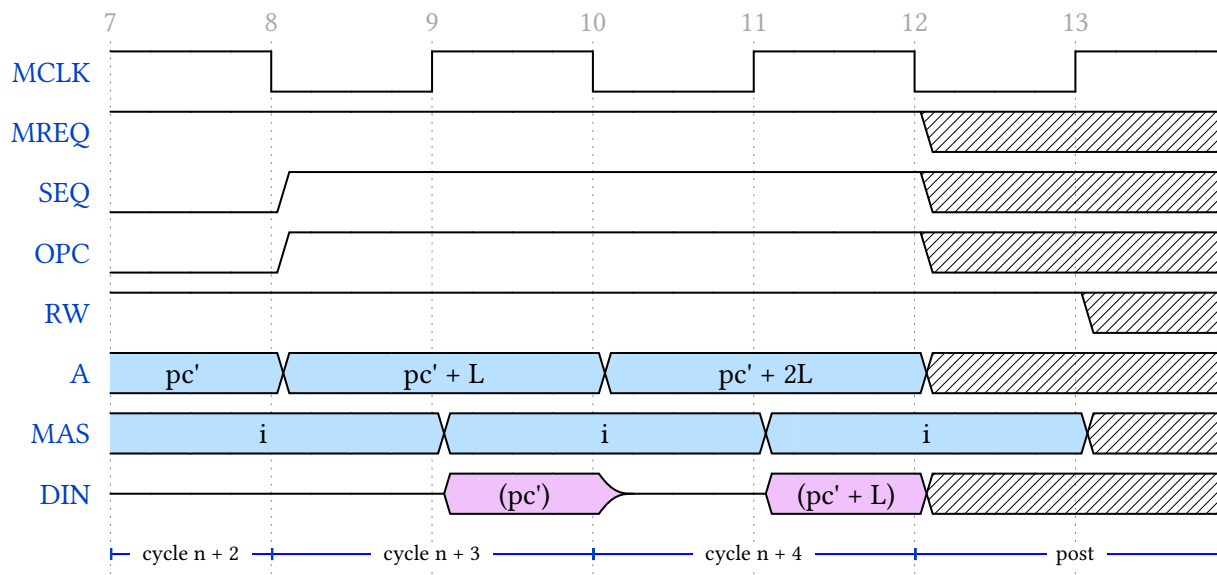


Figure 54: Load Multiple Register Instruction Cycle (n registers including pc) (continued)

## 5.26. STORE MULTIPLE REGISTER INSTRUCTION CYCLE (SMRI-CYCLE)

### RELATED INSTRUCTIONS

STM

### RELATED PROCEDURES

test\_SMRI\_cycle

gba\_request\_SMRI\_cycle

memory\_respond\_SMRI\_cycle

**DURATION:** 2 TO 3 CLOCK CYCLES

	PARAMETERS
pc	Program counter, before executing the instruction.
L	Instruction length, 4 for ARM state, 2 for Thumb state.
alu	The instruction operand—ie, the first target address.
i	MAS, 2 for ARM state, 1 for Thumb state.
s	MAS variable ( BYTE , HALFWORD , or WORD ).
R[k]	The value in the $k$ -th register.

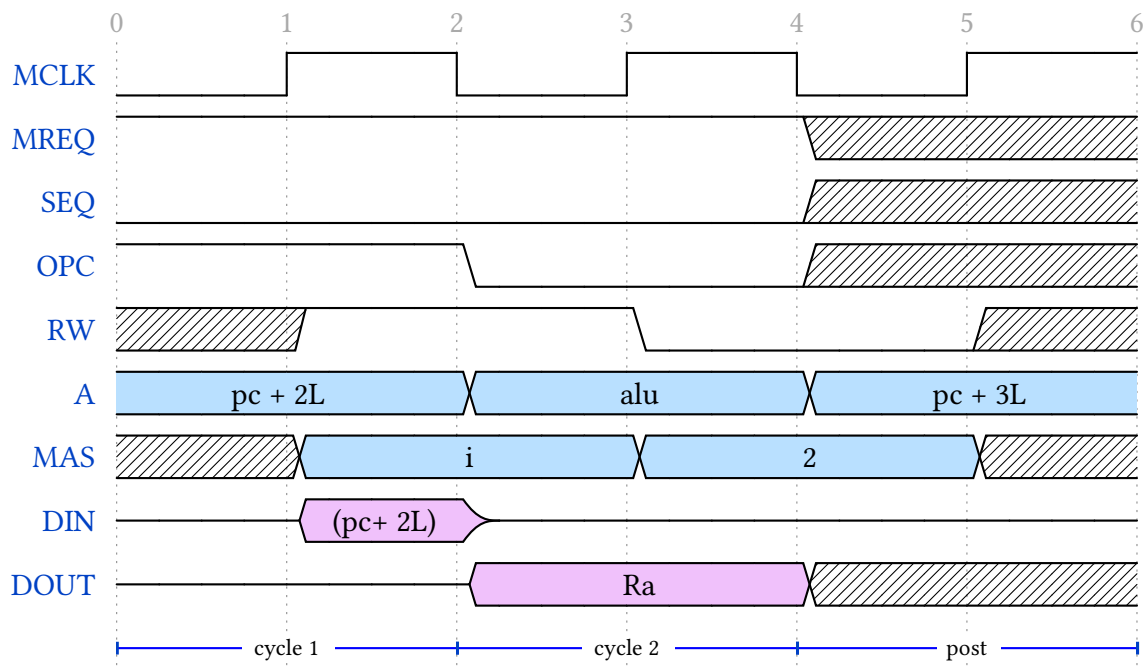


Figure 55: Store Multiple Register Instruction Cycle (single register)

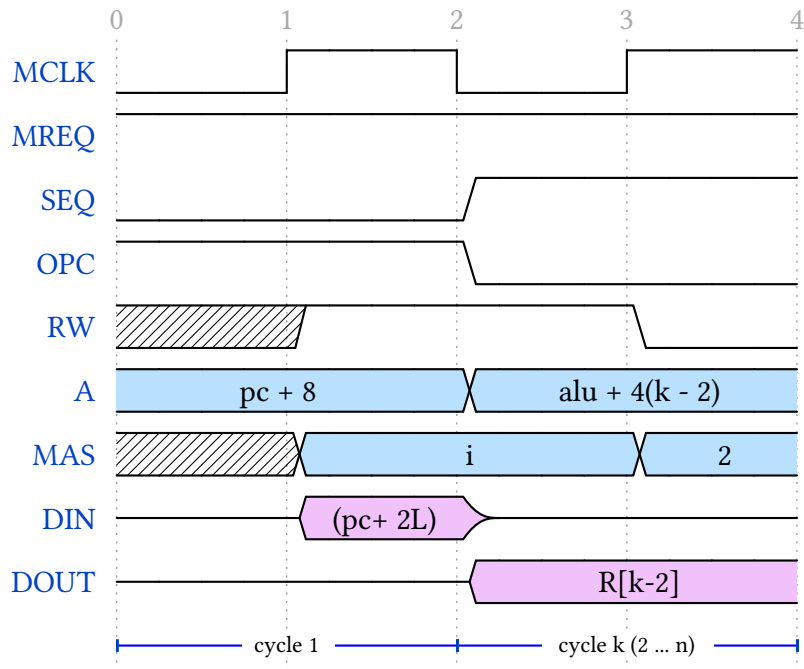


Figure 56: Store Multiple Register Instruction Cycle (n registers)

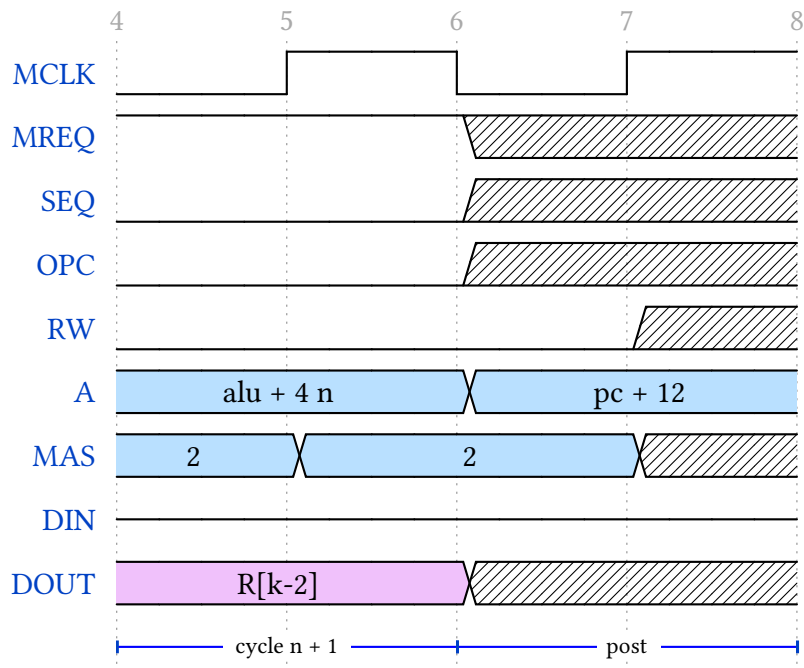


Figure 57: Store Multiple Register Instruction Cycle (n registers) (continued)

## 5.27. DATA SWAP INSTRUCTION CYCLE (DSI-CYCLE)

### RELATED INSTRUCTIONS

SWP , SWPB

### RELATED PROCEDURES

test\_DSI\_cycle

gba\_request\_DSI\_cycle

memory\_respond\_DSI\_cycle

**DURATION:** 4 CLOCK CYCLES

### PARAMETERS

pc	Program counter, before executing the instruction.
alu	The instruction operand—ie, the first target address.
s	MAS variable ( BYTE or WORD ).
R[k]	The value in the $k$ -th register.

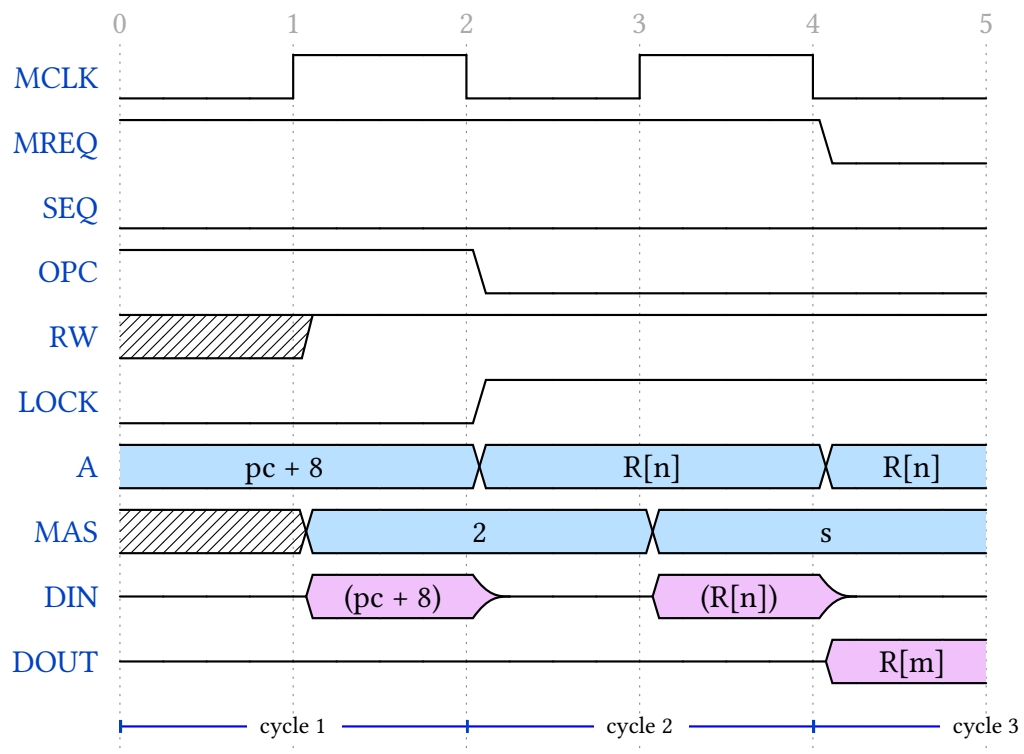


Figure 58: Data Swap Instruction Cycle

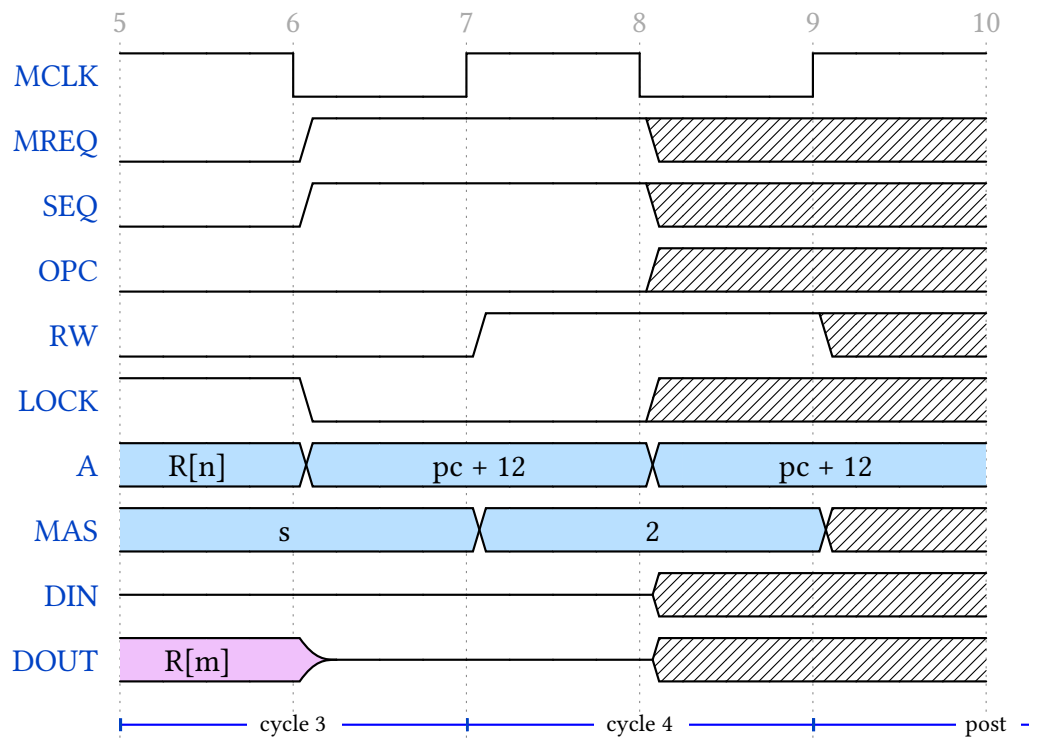


Figure 59: Data Swap Instruction Cycle (continued)

## 5.28. SOFTWARE INTERRUPT AND EXCEPTION INSTRUCTION CYCLE (SIAEI-CYCLE)

### RELATED INSTRUCTIONS

SWI

### RELATED PROCEDURES

test\_SIAEI\_cycle

gba\_request\_SIAEI\_cycle

memory\_respond\_SIAEI\_cycle

**DURATION:** 3 CLOCK CYCLES

	PARAMETERS
pc	Program counter, before executing the instruction.
L	Instruction length, 4 for ARM state, 2 for Thumb state.
old	The processor mode, before executing the instruction
i	MAS, 2 for ARM state, 1 for Thumb state.
T	TBIT, before executing the instruction.
Xn	The exception address.

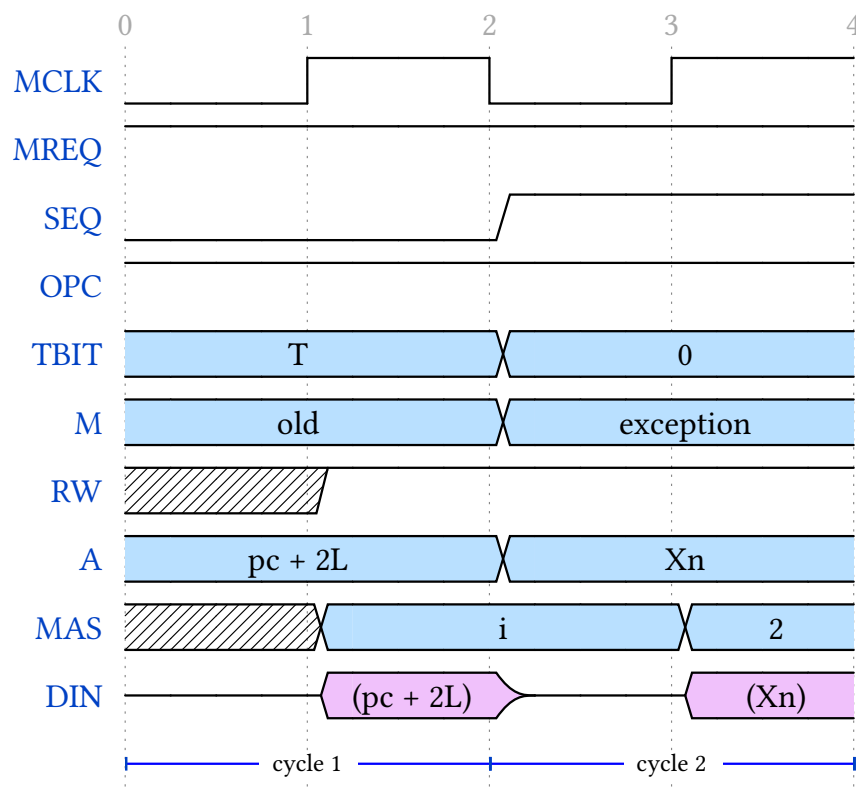


Figure 60: Software Interrupt and Exception Instruction Cycle



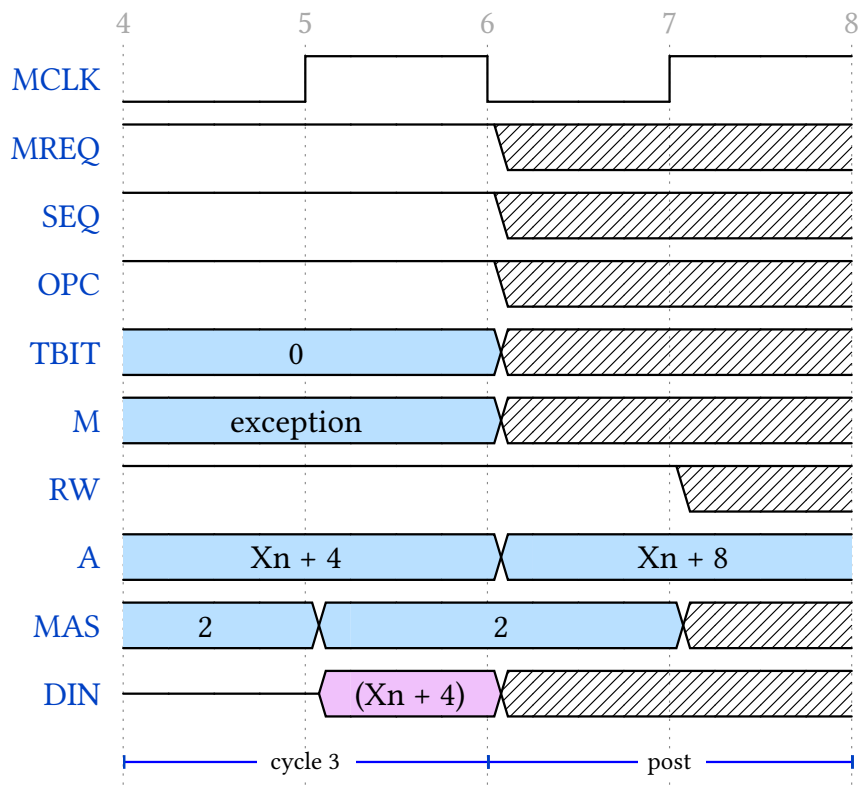


Figure 61: Software Interrupt and Exception Instruction Cycle (continued)

## 5.29. UNDEFINED INSTRUCTION CYCLE (UDI-CYCLE)

### RELATED PROCEDURES

test\_UDI\_cycle

gba\_request\_UDI\_cycle

memory\_respond\_UDI\_cycle

**DURATION:** 4 CLOCK CYCLES

	PARAMETERS
pc	Program counter, before executing the instruction.
L	Instruction length, 4 for ARM state, 2 for Thumb state.
old	The processor mode, before executing the instruction
i	MAS, 2 for ARM state, 1 for Thumb state.
T	TBIT, before executing the instruction.
Xn	The exception address.

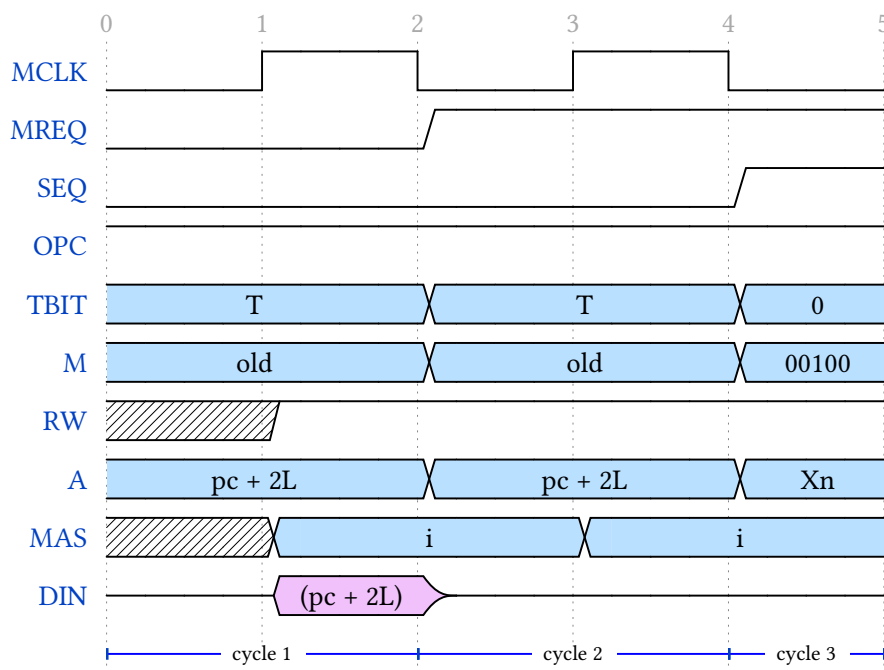


Figure 62: Undefined Instruction Cycle

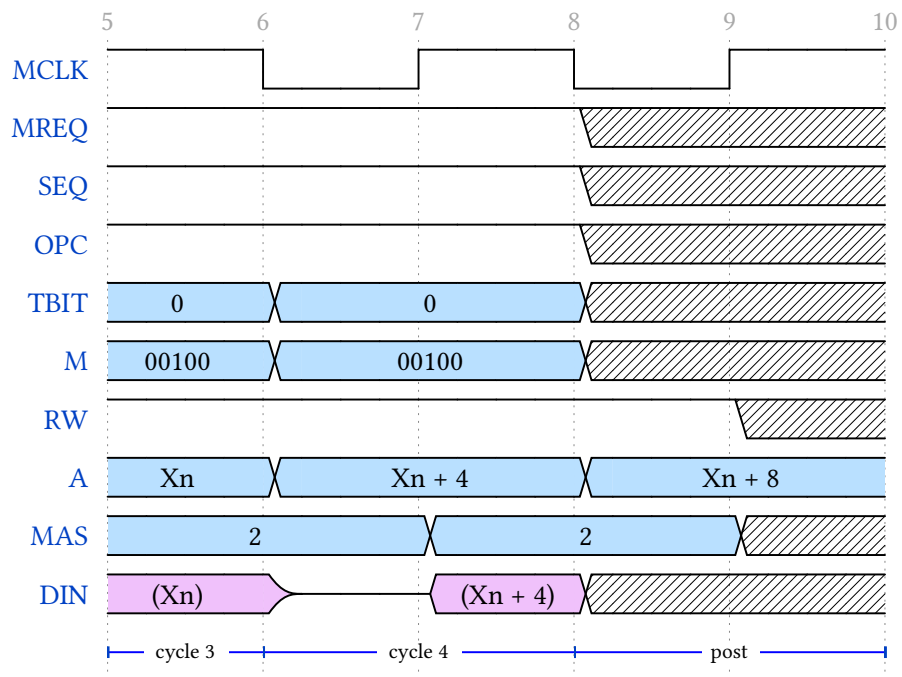


Figure 63: Undefined Instruction Cycle (continued)

### 5.30. UNEXECUTED INSTRUCTION CYCLE (UEI-CYCLE)

#### RELATED PROCEDURES

test\_UEI\_cycle

gba\_request\_UEI\_cycle

memory\_respond\_UEI\_cycle

**DURATION:** 1 CLOCK CYCLE

#### PARAMETERS

**pc** Program counter, before executing the instruction.

**L** Instruction length, 4 for ARM state, 2 for Thumb state.

**i** MAS, 2 for ARM state, 1 for Thumb state.

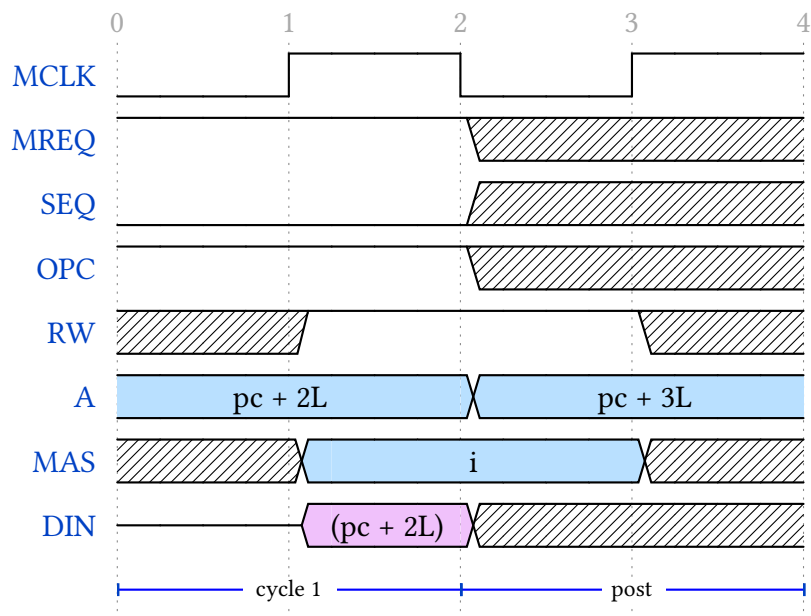


Figure 64: Unexecuted Instruction Cycle

## 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 .....	13
Figure 6	Depipelined Addressing timing diagram .....	14
Figure 7	Data Write Sequence timing diagram .....	15
Figure 8	Data Read Sequence timing diagram .....	16
Figure 9	Halfword-Wide Read Memory Sequence timing diagram .....	17
Figure 10	Halfword-Wide Write Memory Sequence timing diagram .....	18
Figure 11	Byte-Wide Read Memory Sequence timing diagram .....	19
Figure 12	Byte-Wide Write Memory Sequence timing diagram .....	20
Figure 13	Reset Sequence timing diagram .....	21
Figure 14	Reset Sequence timing diagram (continued) .....	21
Figure 15	General timing diagram .....	22
Figure 16	Address Bus Control timing diagram .....	23
Figure 17	Address Bus Control timing diagram .....	23
Figure 18	Data Bus Control timing diagram .....	24
Figure 19	Exception Control timing diagram .....	25
Figure 20	Address Pipeline Control timing diagram .....	26
Figure 21	General Instruction Cycle timing diagram .....	27
Figure 22	Branch and Branch with Link Instruction Cycle .....	28
Figure 23	Branch and Branch with Link Instruction Cycle (continued) .....	29
Figure 24	Thumb Branch with Link Instruction Cycle .....	30
Figure 25	Thumb Branch with Link Instruction Cycle (continued) .....	31
Figure 26	Branch and Exchange Instruction Cycle .....	32
Figure 27	Branch and Exchange Instruction Cycle (continued) .....	33
Figure 28	Data Processing Instruction Cycle (normal) .....	34
Figure 29	Data Processing Instruction Cycle (dest=pc) .....	35
Figure 30	Data Processing Instruction Cycle (dest=pc) (continued) .....	35
Figure 31	Data Processing Instruction Cycle (shift(RS)) .....	36
Figure 32	Data Processing Instruction Cycle (shift(Rs) dest=pc) .....	36
Figure 33	Data Processing Instruction Cycle (shift(Rs) dest=pc) (continued) .....	37
Figure 34	Multiply Instruction Cycle .....	38
Figure 35	Multiply Instruction Cycle (continued) .....	39
Figure 36	Multiply Accumulate Instruction Cycle .....	39
Figure 37	Multiply Accumulate Instruction Cycle (continued) .....	40
Figure 38	Multiply Long Instruction Cycle .....	40
Figure 39	Multiply Long Instruction Cycle (continued) .....	41
Figure 40	Multiply Accumulate Long Instruction Cycle .....	41
Figure 41	Multiply Accumulate Long Instruction Cycle (continued) .....	42
Figure 42	Load Register Instruction Cycle (normal) .....	43
Figure 43	Load Register Instruction Cycle (normal) (continued) .....	44

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