Stack Pointer

The stack is used to store temporary data, local variables, and the return address of interrupts and subroutine calls. Note that the Stack is implemented as growing from higher to lower memory locations. The Stack Pointer Register always points to the top of the Stack. The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. A Stack PUSH command will decrease the Stack Pointer.

The Stack in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. Initial Stack Pointer value equals the last address of the internal SRAM and the Stack Pointer must be set to point above start of the SRAM. Please refer to the system data memory section for the address of the SRAM mapped in the system data store.

Table 2. Stack Pointer Instructions

Instruction	Stack Pointer	Description
PUSH	Decremented 1	Data is pushed onto the stack
CALL ICALL RCALL	Decremented 2	Return address is pushed onto the stack with a subroutine call or interrupt
POP	Incremented 1	Data is popped from the stack
RET RETI	Incremented 2	Return address is popped from the stack with return from subroutine or return from interrupt

The Stack Pointer consists of two 8-bit registers in I/O Memory. The actual length of the Stack Pointer is related to the system implementation.

Register Definition - SPH & SPL

SPH/SPL Stack Pointer Register		
SPH: 0x3E (0x5E) SPL: 0x3D (0x5D)		Default: RAMEND
SP	SP[15:0]	
R/W	R/W	
Bit Definition		
[7:0]	SPL	Stack pointer low 8 bits
[15:8]	SPH	Stack pointer high 8 bits

Instruction Execution Timing

The LGT8XM is driven by the CPU clock (clkCPU), which comes directly from the clock source selection circuit.

Figure 4 on page 14 shows the parallel instruction fetch and execution timing. This is based on Harvard architecture with the fast access register file concept. This is the basic pipelining concept that enables the core to achieve 1 MIPS/MHz execution efficiency.