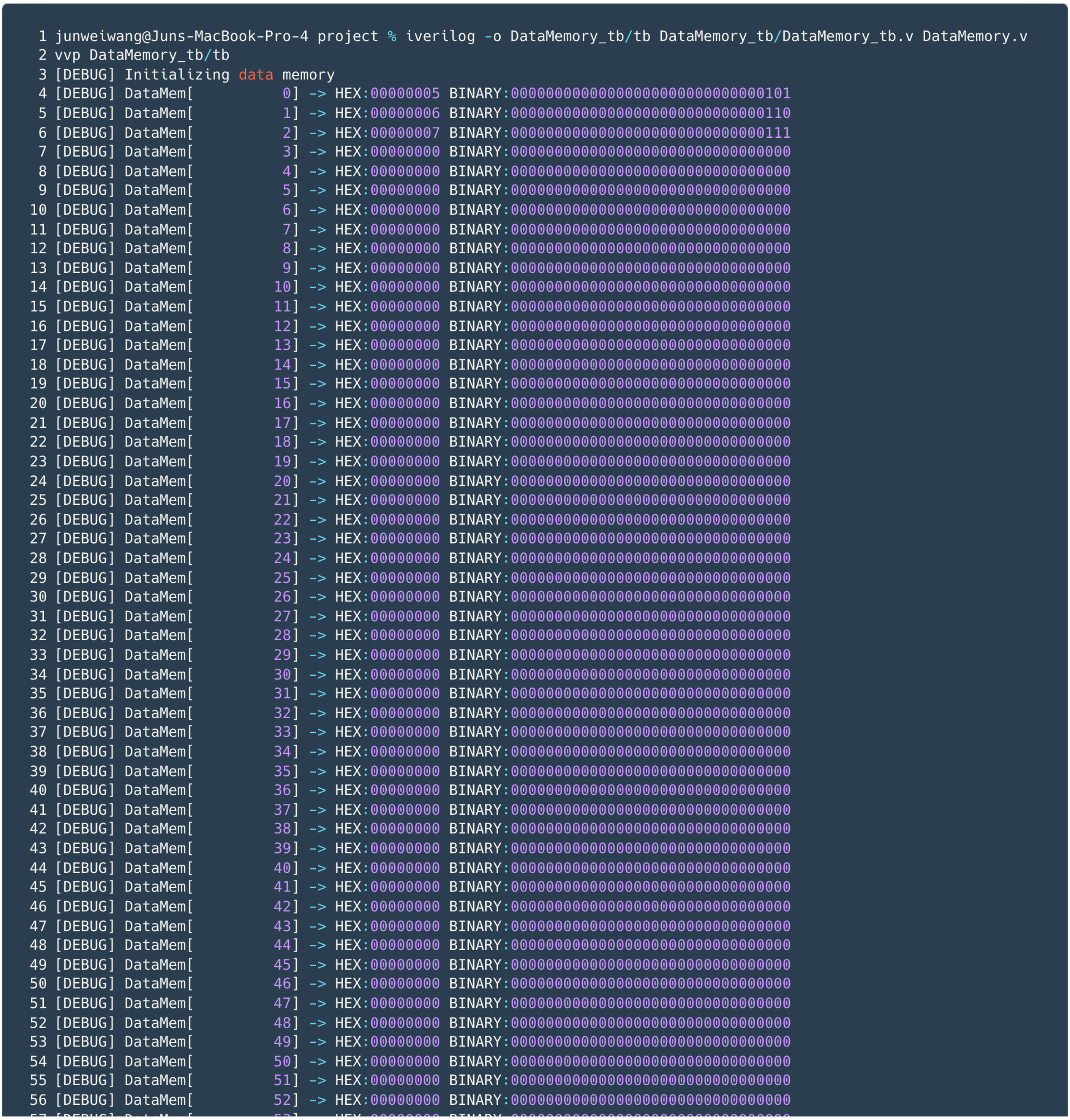
DataMemory

```
iverilog -o tb_DataMemory/tb tb_DataMemory/tb_DataMemory.v DataMemory.v
vvp tb_DataMemory/tb

# open gtkwave to view the waveform
gtkwave tb_DataMemory/tb_data_memory.vcd &
```

Benchmark (verifying the correctness)

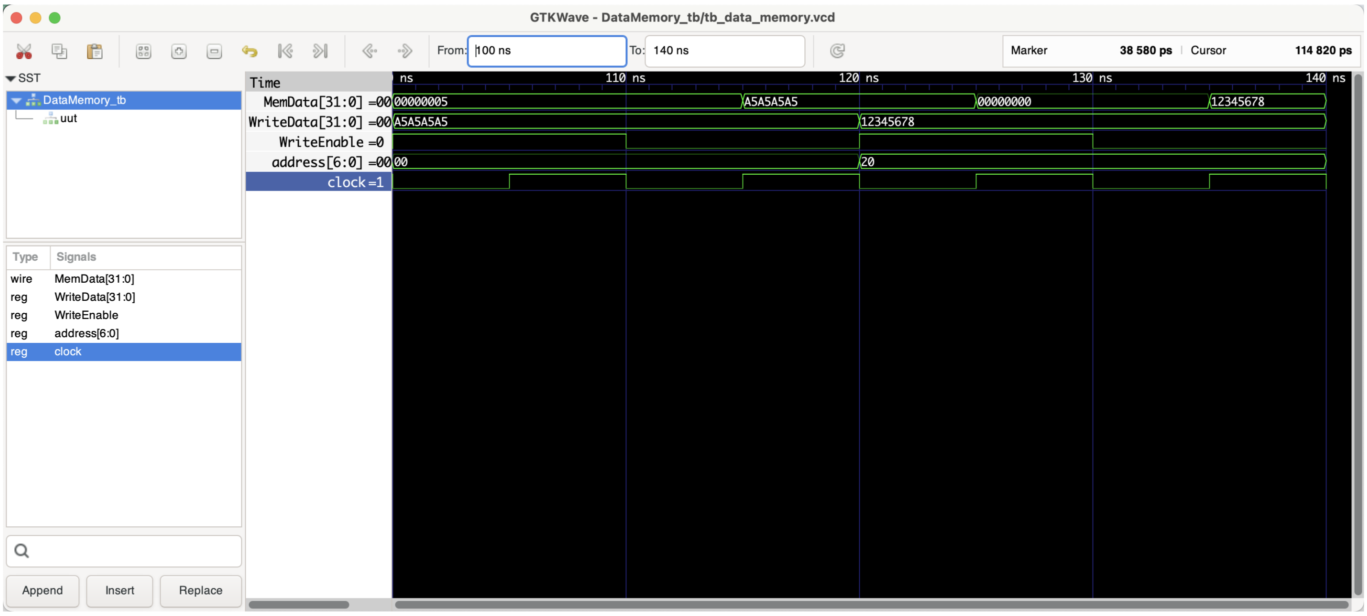
Shell output



```
junweiwang@Juns-MacBook-Pro-4 project % iverilog -o DataMemory_tb/tb DataMemory_tb/DataMemory_tb.v DataMemory.v
vvp DataMemory_tb/tb
[DEBUG] Initializing data memory
[DEBUG] DataMem[         0] -> HEX:00000005 BINARY:00000000000000000000000000000101
[DEBUG] DataMem[         1] -> HEX:00000006 BINARY:00000000000000000000000000000110
[DEBUG] DataMem[         2] -> HEX:00000007 BINARY:00000000000000000000000000000111
[DEBUG] DataMem[         3] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[         4] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[         5] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[         6] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[         7] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[         8] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[         9] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        10] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        11] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        12] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        13] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        14] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        15] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        16] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        17] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        18] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        19] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        20] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        21] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        22] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        23] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        24] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        25] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        26] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        27] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        28] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        29] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        30] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        31] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        32] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        33] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        34] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        35] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        36] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        37] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        38] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        39] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        40] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        41] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        42] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        43] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        44] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        45] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        46] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        47] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        48] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        49] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        50] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        51] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        52] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        53] -> HEX:00000000 BINARY:00000000000000000000000000000000
```

```
[DEBUG] DataMem[        53] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        54] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        55] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        56] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        57] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        58] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        59] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        60] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        61] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        62] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        63] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        64] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        65] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        66] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        67] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        68] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        69] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        70] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        71] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        72] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        73] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        74] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        75] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        76] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        77] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        78] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        79] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        80] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        81] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        82] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        83] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        84] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        85] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        86] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        87] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        88] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        89] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        90] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        91] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        92] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        93] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        94] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        95] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        96] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        97] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        98] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[        99] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       100] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       101] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       102] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       103] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       104] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       105] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       106] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       107] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       108] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       109] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       110] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       111] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       112] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       113] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       114] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       115] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       116] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       117] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       118] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       119] -> HEX:00000000 BINARY:00000000000000000000000000000000
[DEBUG] DataMem[       120] -> HEX:00000000 BINARY:00000000000000000000000000000000Enable = 0, WriteData = a5a5a5a5, MemData = a5a5a5a5
At time              120000, Address = 20, WriteEnable = 1, WriteData = 12345678, MemData = a5a5a5a5
At time              125000, Address = 20, WriteEnable = 1, WriteData = 12345678, MemData = 00000000
At time              130000, Address = 20, WriteEnable = 0, WriteData = 12345678, MemData = 00000000
At time              135000, Address = 20, WriteEnable = 0, WriteData = 12345678, MemData = 12345678
DataMemory_tb/DataMemory_tb.v:58: $finish called at 140000 (1ps)
junweiwang@Juns-MacBook-Pro-4 project %
```

Waveform



Documentation

Module Declaration:

```
module DataMemory (clock, address, WriteEnable, WriteData, MemData);
```

This line declares the module's name `DataMemory` and specifies its interface, which consists of a clock input, an address input, a write enable signal, a data input for writing, and a data output for reading.

Port Definitions:

- `input clock;`: The clock signal input. Memory operations are synchronized with the positive edge of this clock.
- `input WriteEnable;`: A signal that controls whether data should be written to the memory. When high (1), the module will perform a write operation.
- `input [6:0] address;`: A 7-bit address input that specifies the memory address where data should be read from or written to. Only the upper 5 bits (`[6:2]`) are used to index memory, as explained in the comments within the module.
- `input [31:0] WriteData;`: A 32-bit input that contains the data to be written into the memory location specified by the address input when write operations are enabled.
- `output reg [31:0] MemData;`: A 32-bit register that stores the data read from the memory at the specified address.

Memory Declaration:

```
reg [31:0] Mem[0:127]; //32 bits memory with 128 entries
```

This line declares a 128-entry array of 32-bit registers, representing the memory. Each entry corresponds to a word of memory.

Initialization:

The initial block sets all memory entries to 0 and then initializes the first three locations with specific values. It also includes debug messages to display the initial state of the memory.

Functional Description:

- **Write Operation**: On the positive edge of the clock, if `WriteEnable` is high, the module writes the value from `WriteData` into the memory location specified by the upper 5 bits of the `address` input.
- **Read Operation**: On the positive edge of the clock, irrespective of the `WriteEnable` signal, the module updates the `MemData` output with the value from the memory location specified by the upper 5 bits of the `address` input.

Addressing Explanation:

The module uses `address[6:2]` to index the memory array. This technique reflects word-aligned memory addressing commonly used in 32-bit systems. By using only the upper 5 bits of the 7-bit address, the module effectively addresses memory in 4-byte (32-bit) increments, which is consistent with the word size of the memory.

Overall Functionality:

The `DataMemory` module simulates a simple RAM-like memory block in a digital system. It responds to a clock signal for synchronized read and write operations. The memory can be written to when the `WriteEnable` signal is high. It continually provides the data at the current address on the `MemData` output, allowing for data to be read every clock cycle.

Example Usage:

This module could be used in a CPU design as part of the data path, where it would serve as the main data memory, storing and providing access to operands, intermediate results, and other data necessary for program execution. The ability to selectively write or read based on the `WriteEnable` signal allows for versatile memory operation within larger system architectures.