Initialize cache by setting all valid bits to zero.

Open trace file for reading.

Read in command and address from trace file.

WHILE not end-of-file

IF a read command, then // Read Command

IF valid bit is clear, then // Slot Empty (Cache Miss)

Stall CPU.

Read cache line from memory.

Set valid bit.

Clear dirty bit.

Write tag bits.

Deliver data to CPU. // Don’t really do this

Increment MEM\_REF.

Increment CACHE\_MISS.

ELSE // Slot Occupied

IF tag bits match, then // Cache Hit

Deliver data to CPU. // Don’t really do this

Increment CACHE\_HIT.

ELSE // Cache Miss

Stall CPU.

IF dirty bit set, then

Write back cache line to memory. // Don’t really do this

Clear dirty bit.

Read cache line from memory.

Write tag bits. // Valid bit already set

Deliver data to CPU. // Don’t really do this

Increment MEM\_REF.

Increment CACHE\_MISS.

ELSE // Write Command

IF valid bit set, then // Slot Occupied

IF tag bits match, then // Cache Hit

Set dirty bit.

Write data to cache.

Increment CACHE\_HIT.

ELSE // Cache Miss

Stall CPU.

IF dirty bit set, then

Write back cache line to memory. // Don’t really do this

Read cache line from memory.

Write tag bits. // Valid bit already set

Write data to cache. // Don’t really do this

Set dirty bit.

Increment MEM\_REF.

Increment CACHE\_MISS.

ELSE // Slot Empty (Cache Miss)

Stall CPU.

Read cache line from memory.

Write tag bits.

Set valid bit.

Write data to cache. // Don’t really do this

Set dirty bit.

Increment MEM\_REF.

Increment CACHE\_MISS.

Read in command and address from trace file.

END WHILE

Close trace file.

END PROGRAM