Universidad Autónoma de Baja California
FAC. DE CS. QUIM. E INGENIERIA
INGENIERIA EN COMPUTACION

PRACTICA 3

Laboratorio de: Microprocesadores y microcontradores

Equipo:

López Madrigal Leonardo

Maestro:
García López Jesús Adán

Tijuana, B. C. 2 Marzo, 2017

Sección de Memoria (Prueba de memoria RAM)

Objetivo:
El alumno hará uso de una técnica de prueba de memoria aplicándolo en un programa de prueba de memoria RAM.

Material:
-Memoria RAM y Latch para T-Juino.

Equipo:
-Computadora Personal
-Tarjeta T-Juino.
-Protoboard
-Compuertas lógica.
-Una Memoria RAM (2K u 8K)

Teoría:
-Algoritmos de prueba para memoria RAM
-Funciones peek() y poke() en arquitectura x86 (16bits)

Algoritmos de prueba para memoria RAM

FCFS(First In First Out): El primero que entra se ejecuta hasta que termina y luego le toca al siguiente.

STR(Short Time Remainder): Se ejecuta el proceso mas corto.

SRTF(Shortest Remaining Time First): Se ejecuta el que menos ciclos pendientes tiene.

Paginación: Consiste en dividir la memoria interna en zonas iguales llamadas frames, y los programas en partes de mismo tamaño denominadas paginas mediante la tabla de paginas asigna las direcciones físicas.

Segmentación: Divide la información en trozos de diferentes tamaños.

Swapping: Carga los procesos primero en el swap y luego en la memoria principal mediante swap – in y swap-uot.

Fragmentacion: Es un método para aprovechar el espacio, cuando no sobre

Externa: El proceso entra pero sobre espacio.

Memtest86+ escribe una serie de patrones de prueba a cada dirección de memoria, y luego lee los datos comparándolos a la búsqueda de errores.

La información acerca del chipset se puede usar para mejorar estas pruebas, especialmente en sistemas que utilizan overlock. Muchos chipsets pueden informar acerca de la velocidad de la RAM, y alguno permite el cambio de esta velocidad dinámicamente; de esta manera, con Memtest86+ se puede comprobar hasta qué punto la memoria continúa sin errores si subimos la velocidad.

Tipos de test:

•Test 0: [Address test, walking ones, no cache]. Test de todos los bits direccionables en todos los bancos de memoria usando un patrón de acceso "walking ones".

•Test 1: [Address test, own address]. Cada dirección es escrita con el valor de su propia dirección y luego es probada para detectar diferencias. Este test es complementario y más estricto que el Test 0 y debería detectar todos los errores de direccionamiento.

•Test 2: [Moving inversions, ones&zeros]. Este test utiliza el algoritmo Moving inversions con patrones de unos y ceros. Es un test rápido que solamente da errores en subsistemas de memoria muy dañados.

•Test 3: [Moving inversions, 8 bit pattern]. Utiliza el algoritmo Moving Inversions diseñado para detectar fallos producidos por interferencia con las células de memoria adyacentes.

•Test 4: [Moving inversions, random pattern]. Se utiliza el mismo algoritmo del paso 3 pero el patrón es un número aleatorio (más bien pseudoaleatorio) y su complemento. Es un test muy efectivo para detectar errores de datos, utilizando 60 patrones aleatorios cambiando en cada pasada del test. Por ello múltiples pasadas aumentan la eficacia.

•Test 5: [Block move, 64 moves]. Este test prueba la memoria utilizando la instrucción [movsl] y está basado en un antiguo test llamado burnBX de Robert Redelmeier. Experimentalmente es de los test que revelan errores más sutiles.

•Test 6: [Moving inversions, 32 bit pat]. Es un test bastante lento pero muy efectivo para detectar errores de datos, ya que hace 32 pasadas para probar todos los patrones.

•Test 7: [Random number sequence]. Se escribe una serie de números aleatorios en memoria. Es comprobado y complementado y vuelto a comprobar.

•Test 8: [Modulo 20, ones&zeros]. Utiliza el algoritmo Modulo-X, diseñado para evitar interferencia del subsitema de caché que podrían enmascarar algunos errores en tests anteriores. Utiliza patrones de unos y ceros.

•Test 9: [Bit fade test, 90 min, 2 patterns]. Se inicializa toda la RAM con un patrón y se deja inactiva 90 minutos, entonces se examina en busca de alguna variación. Se pasa dos veces, una con ceros y otra con unos. Dura 3 horas y no forma parte del test standard, hay que seleccionarlo a mano en el menú.

Funciones peek() y poke() en arquitectura x86 (16bits)

Peek():

Es una instrucción, que usaba el lenguaje BASIC de los antiguos ordenadores de 8 bits, como el Sinclair BASIC del ZX Spectrum.

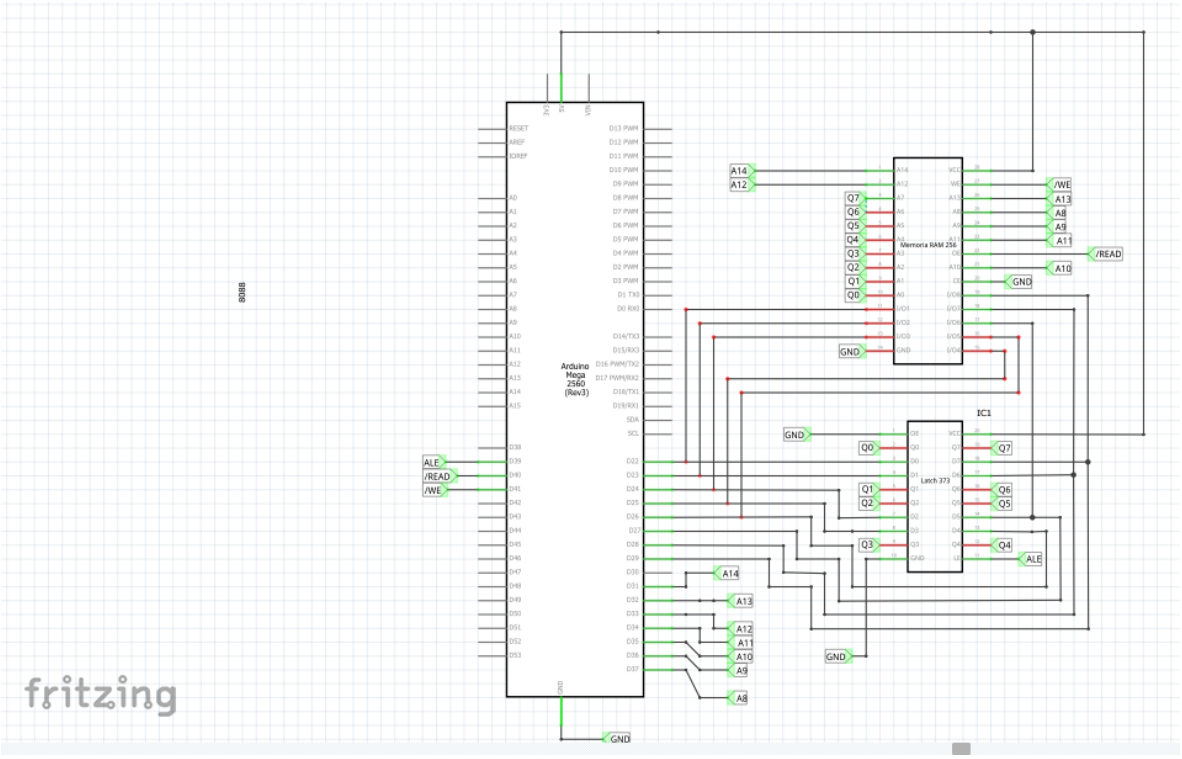
Dicha instrucción se encargaba de recuperar un valor en una determinada dirección (posición) de memoria.

Poke():

Es una instrucción en el lenguaje de programación BASIC usada en algunas de las primeras versiones de este lenguaje. Dicha instrucción se encargaba de grabar un valor en una determinada dirección de memoria, ambos datos dados como argumento.

Las instrucciones PEEK y POKE fueron implementadas por primera vez por Bill Gates y Paul Allen en 1975, en el lenguaje Altair BASIC.

Esquemático



Conclusión:

López Madrigal Leonardo

En la práctica de la memoria RAM consistió en hacer tres programas, uno el cual es ensamblador que fue hacer un mapeo de memoria en la memoria interna del T-juino (estresar los bits) para saber si ciertas secciones de memoria funcionan en el labaratorio, y el otro fue hacer unas funciones llamas peek y poke para el mapeo de memoria pero con un ram externa la cual iniciaba en otra posición de memoria de la ram externa esta se hizo una prueba en el laboratorio , esto fue posible alambrando el 8088 con un latch y la ram externa la cual para eso hice un esquemático en fritzing para no batallar.