PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA
INGENIERÍA INFORMÁTICA

INFORME: EMULADOR DEL PROCESADOR 6502

HORARIO: 0681

GRUPO: 9

INTEGRANTE:

Carranza Cordova, Gerson Nick - 20134914

PROFESOR:

Jimenez Garay, Gabriel Alexandro

ÍNDICE

1. Estructuras Utilizadas

El emulador del procesador Mos6502 hace uso de dos estructuras esenciales para simular la arquitectura del procesador y una estructura auxiliar para poder manejar, de manera más sencilla, el registro de estados.

Por un lado, la primera estructura esencial, “Mos6502”, está conformada por 8 miembros: 6 uint8_t y 2 uint16_t. Los miembros del tipo “uint8_t” son A (Register A), X (Register X), Y (Register Y), SP (Stack Pointer), PSR (Program Status Register) e Inst (Instruction Register) y los dos miembros del tipo “uint16_t” son PC (Program Counter) y Addr (Address Bus). Los miembros A, X, Y, PSR, Inst, Addr son inicializados en 0 por motivo de limpieza de datos; SP es inicializado en 0xFF, ya que la pila hace uso de la página 0x01 (rango 0x0100 – 0x01FF) y crece desde el espacio de memoria más alto hacia el más bajo; y el PC es inicializado en el valor combinado de los espacios de memoria 0xfffd y 0xfffc de la RAM.

Por otro lado, la siguiente estructura esencial, “Memory”, está conformada por 1 miembro: un puntero a uint8_t llamado ram. Este miembro apuntará a un espacio de memoria de 64K conformado por un arreglo de uint8_t.

Por último, la estructura auxilizar, “FlagMask” consta de 8 miembros del tipo uint8_t llamados carry, zero, int_disable, decimal_mode, break_cmd, expansion, overflow, negative. Cada miembro representa el bit que le corresponde en un byte para hacer referencia a que se activó. Por ejemplo, carry hace referencia a b00000001, zero hace referencia a b00000010 y así sucesivamente.

2. Implementación de las etapas del pipeline

Las etapas del pipeline del procesador Mos 6502 fueron representadas por tres funciones: fetch, decode y execute. Estas funciones toman como parámetros un puntero a la estructura Mos6502 y otro puntero a la estructura Memory.

La función “fetch” asigna al “registro inst” (cpu->inst) el valor de la posición de la memoria ram (mem->ram[cpu->pc]) dada por el “registro pc” (cpu->pc). Finalmente, le agrega una unidad al valor del registro pc (cpu->pc ++).

Primero, la función “decode” obtiene un puntero a una función que representa el modo de direccionamiento respectivo. Este puntero es obtenido gracias a que el modo de direccionamiento se puede extraer directamente de los bits de la instrucción. La mayoría de instrucciones están conformadas por un patrón de la forma “aaabbbcc”. Las secciones “aaa” y “cc” determinan el opcode mientras que la sección “bbb” determina el modo de direccionamiento. Para cada combinación de “cc” de la forma 00, 01, 10 existen diferentes combinaciones de modo de direccionamiento. Por ejemplo, para “cc = 10 “, “bbb = 000” representa el modo de direccionamiento inmediato, “bbb = 001” representa el modo de direccionamiento zero page y así sucesivamente. No existen instrucciones para la combinación de “cc” de la forma 11. Luego de obtener el modo de direccionamiento se procede a ejecutarlo (la lógica de cada modo de direccionamiento será explicada

posteriormente) dando como resultado ya haber leído los operandos correspondientes al modo de direccionamiento.

Finalmente, la función “execute” obtiene un puntero a la función que representa la instrucción que es requerida en ese momento. Este puntero, simplemente, se obtiene comparando el valor del registro “inst” con el o los valores que posee una instrucción sin tomar en cuenta el modo de direccionamiento, ya que este fue obtenido anteriormente. Luego de obtener el puntero a la función que representa la instrucción esta es ejecutada terminando asi la función “execute”.

3. Prototipos utilizados en las instrucciones

Todas las funciones que representan a las instrucciones del procesador Mos 6502 reciben dos parámetros: un puntero a la estructura “Mos6502” y otro a la estructura “Memory”. La mayoría de estas hace uso de operaciones aritméticas que fueron implementadas con funciones del lenguaje de programación c y también, según la instrucción, modificarán el program status register.

Algunos ejemplos de los prototipos de las instrucciones se mostrarán a continuación

```
void ADC(Mos6502 *cpu, Memory *mem);
void AND(Mos6502 *cpu, Memory *mem);
void ASL(Mos6502 *cpu, Memory *mem);
void BCC(Mos6502 *cpu, Memory *mem);
void BCS(Mos6502 *cpu, Memory *mem);
void BEQ(Mos6502 *cpu, Memory *mem);
void BIT(Mos6502 *cpu, Memory *mem);
void BMI(Mos6502 *cpu, Memory *mem);
void BNE(Mos6502 *cpu, Memory *mem);
void BPL(Mos6502 *cpu, Memory *mem);
void BRK(Mos6502 *cpu, Memory *mem);
```

Figura 1: Prototipos de instrucciones

4. Prototipos utilizados en los modos de direccionamiento

Las funciones que representan a los modos de direccionamiento, al igual

que el caso de las instrucciones, reciben dos parámetros: un puntero que apunta a la estructura “Mos6502” y otro puntero que apunta a la estructura “Memory”. Estas funciones trabajan con el bus de direcciones y con la posición de la memoria ram dada por el valor del contador del programa. Según el modo de direccionamiento, leerán cero, una o dos veces

datos de la memoria ram para combinarlos y asignarlos al bus de datos. Cada lectura de estos datos representará un aumento del contador del programa.

A continuación se mostrarán los prototipos de las funciones que representan los modos de direccionamiento.

```
void accumulator(Mos6502 *cpu, Memory *mem);
void absolute(Mos6502 *cpu, Memory *mem);
void absolute_x(Mos6502 *cpu, Memory *mem);
void absolute_y(Mos6502 *cpu, Memory *mem);
void immediate(Mos6502 *cpu, Memory *mem);
void implied(Mos6502 *cpu, Memory *mem);
void indirect(Mos6502 *cpu, Memory *mem);
void x_indirect(Mos6502 *cpu, Memory *mem);
void indirect_y(Mos6502 *cpu, Memory *mem);
void relative(Mos6502 *cpu, Memory *mem);
void zeropage(Mos6502 *cpu, Memory *mem);
void zeropage_x(Mos6502 *cpu, Memory *mem);
void zeropage_y(Mos6502 *cpu, Memory *mem);
```

Figura 2: Prototipos de modos de direccionamiento

5. Información adicional

Se implementaron dos funciones llamadas “show_cpu_info” y “execute_program” adicionales. La primera función muestra información relevante sobre los valores de la estructura Mos6502 y la segunda función itera sobre los valores de la ram hasta que encuentre un null (todo el programa) ejecutando las funciones fetch, decode, execute y show_cpu_info por cada instrucción.

Link del repositorio: https://github.com/dralibih/mos6502