Modos de direccionamiento y control de flujo

Área de Arquitectura y Tecnología de Computadores – Versión 1.0.899, 18/02/2022

Objetivos

En esta sesión el alumno codificará y ejecutará, utilizando el simulador del Computador Teórico, dos tipos de instrucciones, para lograr, en cada caso, diferentes objetivos:

- Instrucciones de movimiento de datos: con el objetivo de comprender y utilizar todos los modos de direccionamiento que la CPU pone a disposición del programador, y la relación entre estos modos y el código máquina de las instrucciones correspondientes.
- Instrucciones de control de flujo: con el objetivo de comprender cómo el flujo secuencial de la ejecución puede ser roto mediante los saltos, cómo se codifican éstos y cómo su ejecución afecta al registro `PC`.

Conocimientos y materiales necesarios

Para poder realizar esta sesión, el alumno debe:

- Acudir al laboratorio con una copia del juego de instrucciones de la CPU. El alumno también debe conocer perfectamente el manejo de la tabla de codificación de las instrucciones de esta CPU.
- Repasar el juego de instrucciones del Computador Teórico y conocer todos los tipos de instrucciones, los modos de direccionamiento posibles y el trabajo que realiza cada instrucción.
- Durante la sesión se plantearán una serie de preguntas que puedes responder en el correspondiente cuestionario en el Campus Virtual. Puedes abrir el cuestionario en otra pestaña del navegador pinchando en el enlace mientras mantienes pulsada la tecla `Ctrl`.

Desarrollo de la práctica

A continuación iremos codificando y ejecutando una serie de instrucciones que usarán diferentes modos de direccionamiento y saltos. Para seguir la pista de dónde está almacenado el código máquina de cada instrucción y qué instrucción representa, se recomienda crear en un papel aparte una tabla como la siguiente:

Dirección de memoria	Código Máquina	Mnemónico Instrucción
...	...	...
...	...	...

que irás rellenando a lo largo de esta sesión de prácticas, cada vez que se te pida codificar una instrucción.

1. Modos de direccionamiento

Vamos a estudiar los distintos modos de direccionamiento de la CPU. En la sesión de prácticas anterior ya has usado un modo de direccionamiento (instrucciones `mov r5, r3` y `add r0, r5, r3`) ¿Cómo se denomina este modo de direccionamiento?

1.1. Preparación del simulador

- Tras arrancar el Simulador (si ya lo tenías arrancado, reinícialo), modifica los siguientes registros (pulsando con el ratón sobre ellos) y carga los valores que se especifican:

Registro	Valor
r2	FC23h
r3	0004h
r7	ABCDh
PC	0200h

- Además, carga el número `5004h` en la dirección de memoria `0004h`.

1.2. Codificación y ejecución de instrucciones

- Codifica la instrucción `mov r5, [r3]`. Carga el código de la instrucción en la memoria, de tal manera que sea la próxima instrucción a ejecutar. Apúntalo en la tabla que estás creando en papel aparte. No pulses `F8` todavía. Responde antes a las siguientes cuestiones:
  1. ¿Qué es lo que hará esta instrucción?
  2. ¿Con qué operandos trabaja?
  3. ¿Qué valor aparecerá en el registro `r5` después de que se ejecute la instrucción? Responde en el cuestionario: pregunta 1. ¿Y en `r3`? Responde en el cuestionario: pregunta 2.
  4. ¿Cómo se denomina el modo de direccionamiento utilizado en esta instrucción?

  Pulsa `F8` y comprueba si ha ocurrido lo que habías previsto.
- Ahora deberás escribir las instrucciones necesarias para cargar en el registro `r2` el valor `0020h`. ¿Cuantas instrucciones necesitas? ¿Por qué? Codifícalas y cárgalas en memoria para que sean las siguientes instrucciones a ejecutar (a partir de la situación anterior). Pulsa `F8` tantas veces como sea necesario para que se cargue el citado valor en el registro `r2`. ¿Cómo se llama el modo de direccionamiento que has usado? ¿Por qué recibe ese nombre?
- Ahora tienes que guardar el contenido del registro `r7` en la posición de memoria `0020h`. ¿Qué instrucción tienes que utilizar? Responde en el cuestionario: pregunta 3. ¿Qué modo de direccionamiento hay que aplicar? Codifica la instrucción, cárgala en la memoria y ejecútala. Comprueba que, efectivamente, el dato se ha almacenado en la dirección deseada (usando para ello el editor hexadecimal de la memoria).

2. Instrucciones de control de flujo

Vamos a trabajar ahora con las instrucciones que permiten controlar el flujo de ejecución de un programa. No reinicies el simulador.

2.1. Salto incondicional

- Tienes que cargar en memoria, en la dirección correspondiente a la próxima instrucción a ejecutar, una instrucción que cause un salto hacia atrás, de modo que se vuelva a la primera instrucción de esta sesión (`mov r5, [r3]`). Consulta la tabla en la que vas apuntando las instrucciones y la posición de memoria en las que éstas se encuentran. Responde a las siguientes cuestiones:
  1. ¿Cuál es el mnemónico de la instrucción de salto incondicional necesaria? Responde en el cuestionario: pregunta 4.
  2. ¿Cuál es el código máquina de esa instrucción? Responde en el cuestionario: pregunta 5. Cárgalo en memoria.
  3. Pulsa `F8` (sólo una vez). ¿Qué valor tiene ahora el registro `PC`? ¿Es el esperado? (Si ese valor no es `0200h` has realizado mal el cálculo). Si no lo has hecho bien piensa en qué te has equivocado. Si lo has hecho mal, puedes reintentar hacerlo bien cargando en el registro `PC` el valor anterior (`0204h`) y modificando la instrucción de salto incondicional.
- Si ahora sustituimos la instrucción `mov r5, [r3]` por la instrucción `jmp +3`, y luego pulsamos 7 veces `F8`, ¿cuántas instrucciones *distintas* se ejecutarán? Responde en el cuestionario: pregunta 6. ¿Cuál será el valor final del registro `PC`? Responde en el cuestionario: pregunta 7. ¿Por qué? Compruébalo.

2.2. Salto condicional

Pasaremos ahora a las instrucciones de salto condicional, pero en lugar de modificar la memoria "a mano", usaremos un *archivo de memoria*. El formato de los archivos de memoria ha sido explicado en la sesión anterior.

- Selecciona la opción del menú **Ejecución › Reiniciar**. Esto borrará todos los registros y la memoria.
- Edita un archivo de memoria llamado `4-2prog1.mem` que contenga el código máquina de un programa que debe hacer lo siguiente:
  1. Cargar el valor 8 decimal en el registro `r3` (se puede hacer con una o con dos instrucciones ¿Por qué? Hazlo con dos instrucciones).
  2. Cargar el valor 2 decimal en el registro `r6`. Hazlo también en dos instrucciones.
  3. Decrementar el contenido del registro `r3` en una unidad.
  4. Ejecutar la instrucción `brnz -2`.
  5. Sumar el contenido de los registros `r3` y `r6` y guardar el resultado en el registro `r7`.
- Carga el archivo de memoria en el simulador a partir de la dirección `0300h` e inicializa el registro `PC` con el valor `0300h`. De este modo la ejecución comenzará en la primera instrucción de las que has cargado en memoria. Comprueba, con el desensamblador, que has codificado correctamente todas las instrucciones.
- Responde a las siguientes cuestiones: ¿cuántas veces tendrás que pulsar `F8` antes de que se ejecute la última instrucción (la suma)? Responde en el cuestionario: pregunta 8. La primera vez que se ejecuta la instrucción `brnz -2`, ¿qué instrucción se ejecutará a continuación? Responde en el cuestionario: pregunta 9. Comprueba, pulsando `F8`, que estás en lo cierto.
- Vuelve a cargar el archivo de memoria `4-2prog1.mem`. En el editor hexadecimal de la memoria, sustituye el código de la instrucción de salto condicional por el código de la instrucción `brns -2`. Responde a las mismas preguntas que en el caso anterior. ¿Hay alguna diferencia?
- Haz lo mismo que en el caso anterior, pero cambiando el código del salto condicional por el código de `brnc -2`. Responde a las mismas preguntas.

2.3. Bucles

Examinemos lo que hemos hecho hasta ahora. En el archivo `4-2prog1.mem` tenemos un conjunto de instrucciones que hacen lo siguiente:

- Cargar en un registro (`r3`) un valor *N* determinado. En este caso *N*=8.
- Decrementar el contenido de ese registro, de uno en uno, hasta cero (cuando la instrucción condicional es `brnz`). Dicho de otra manera, podemos hacer una cuenta atrás desde un número *N* determinado.

Lo que vamos a hacer a continuación es aprovechar la cuenta para hacer alguna operación en cada decremento.

- Copia el archivo de memoria `4-2prog1.mem` en un nuevo archivo llamado `4-2prog2.mem`.
- Edita el archivo `4-2prog2.mem` y, delante del código de la instrucción `dec r3`, añade el código de la instrucción `inc r0`. Cambia el código de la instrucción `brnz -2` por el de la instrucción `brnz -3`. Guarda el contenido del archivo.
- Carga el archivo de memoria y ejecútalo hasta la instrucción `add`. ¿Qué valor tiene el registro `r0`? ¿Coincide con el valor de *N*? Responde en el cuestionario: pregunta 10.

3. Ejercicios adicionales

3.1. Archivos en la carpeta de trabajo

En tu carpeta de prácticas deberás tener los archivos de programa: `4-2prog1.mem` y `4-2prog2.mem`.

3.2. Ejercicios

✓ Considera el programa que has codificado en `4-2prog1.mem`. Si quisieras sustituir la instrucción `dec r3` por una instrucción `sub` que hiciera lo mismo, ¿cuántas instrucciones tendrías que añadir al programa para conseguirlo? Responde en el cuestionario: pregunta 11.

✓ Carga el archivo `4-2prog2.mem` en la memoria (elige tú mismo la dirección que quieras) e inicializa el registro `PC` para apunte a la primera instrucción del programa. Modifica (directamente sobre la memoria) las instrucciones que inicializan `r3` para que se cargue el valor 12 decimal en lugar del 8.

✓ Escribe un nuevo programa (en un archivo de memoria) a partir del que hay en `4-2prog2.mem`, haciendo los cambios necesarios para sustituir la instrucción `inc r0` por una instrucción `add` que tenga el mismo efecto. Cárgalo y ejecútalo para verificar que funciona correctamente.

✓ Reinicia el simulador. Codifica la instrucción `add r3, r5, r1`. Cárgala en una dirección de memoria cualquiera. En la dirección siguiente carga la codificación de la instrucción de salto condicional `brz +5`. Ahora, en la dirección de memoria a la que se saltaría con ese `brz`, carga la codificación de una instrucción de salto incondicional cuya ejecución lleve a la instrucción `add`.

Prepara el registro `PC` para que apunte a la instrucción `add` y carga, a mano, los registros `r5` y `r1` de tal manera que al pulsar `F8` tres veces se ejecuten ambas instrucciones de salto: la condicional y la incondicional.

✓ Cambia la instrucción de salto condicional del ejercicio anterior por todas las posibles condiciones. Carga los registros de manera adecuada para probar cada una de las instrucciones.

✓ Codifica la instrucción `jmp -1`. Cárgala en una dirección cualquiera y modifica el valor del registro `PC` para que sea la próxima instrucción a ejecutar. Pulsa varias veces `F8`, ¿qué sucede?