Universidad de Valladolid
Departamento de Informática

Escuela Universitaria Politécnica c/Francisco Mendizábal,1 47014 Valladolid Tel.:(983) 423501-02 Fax:(983) 423490

Ordenadores

2º I. T. Telecomunicación

Examen ordinario - 5 de febrero de 2.002
Soluciones

Problema 1

Apartado 1

La figura 1 muestra el número de líneas de datos y de dirección que pueden entrar a cada uno de los chips de memoria del problema. Dado que los primeros 64Kpalabras deberán ser de ROM, parece claro que ese espacio de memoria deberá construirse utilizando cuatro módulos de ROM de 16K x 16 bits. Las 448Kpalabras restantes deberán construirse utilizando RAM. Aunque existen diferentes posibilidades, la más razonable parece colocar dos módulos de RAM de 64K x 8 bits a continuación de los módulos de ROM. Entre ambos tipos de memoria se han completado 128K palabras de 16 bits. Una vez hecho esto, se colocan tres módulos de RAM de 128Kpalabras x 16 bits para completar el resto del espacio hasta las 512 Kpalabras.

La figura 2 muestra la distribución de los módulos de memoria, así como la conexión de las líneas del bus de datos y de direcciones. Las líneas de control utilizadas para la lectura (R) y la escritura (W), no representadas en la figura, se conectan como sigue: la línea R va a todos los chips de memoria y la linea W va a todos los chips de memoria RAM. Se utiliza un primer decodificador de 2 a 4 para seleccionar uno de entre cuatro módulos de 128 Kpalabras, haciendo uso de las líneas `a18` y `a17`. Las tres salidas de más peso se conectan a las entradas "chip select" de los tres módulos de RAM de 128 Kpalabras. La salida de menos peso se usa para habilitar un segundo decodificador, de 1 a 2 (también podrían usarse puertas lógicas), que permite seleccionar entre las 64 Kpalabras superiores (las de RAM) o las inferiores (de ROM), haciendo uso de la linea `a16`. Si se activa la salida de menos peso del decodificador, dicha salida habilita a su vez a un tercer decodificador, de 2 a 4, que permite seleccionar uno de los cuatro módulos ROM de 16Kpalabras haciendo uso de las señales de control `a15` y `a14`.

Las líneas de dirección de menos peso entran directamente a cada uno de los chips correspondientes. Puede observarse que todas las líneas de dirección intervienen de una u otra manera en la selección de datos dentro de cada chip.

Apartado 2

La palabra con dirección `0x186a0` es, en binario, la dirección siguiente (representándola en 19 bits):

```
001 1000 0110 1010 0000
```

Puede verse que esta dirección habilita la salida 0 del decodificador inferior, con lo que se activa el decodificador intermedio. A su vez, la línea `a16` está a 1, por lo que se habilita la pareja de RAM de 64K x 8. En consecuencia, el dato está en estos chips. Puede verse también que la dirección `0x186a0` es en decimal la dirección 100.000, lo que se corresponde con el rango de direcciones almacenado en dicha pareja de chips, que va de 64Kpalabras a (128Kpalabras-1).

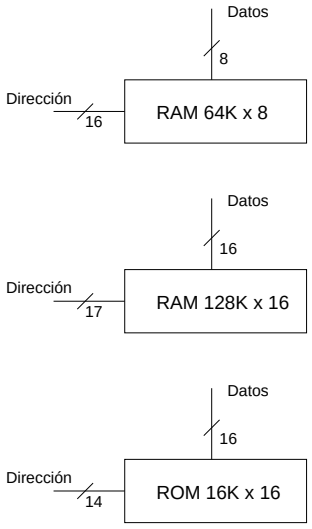

Figura 1: Módulos de memoria del problema 1.

Problema 2

Se comienza por calcular el tamaño de los campos de código de operación para cada uno de los tipos de instrucciones. Se tiene lo siguiente:

- Las instrucciones sin operandos tienen 32 bits de campo de código de operación.
- Las instrucciones con 1 operando tienen 32-24=8 bits de campo de código de operación.
- Las instrucciones con 2 operandos tienen 32-12-12=8 bits de campo de código de operación.
- Las instrucciones con 3 operandos tienen 32-6-6-4=16 bits de campo de código de operación.

Por lo tanto, se comienza por colocar las instrucciones de 1 y de 2 operandos, ya que tienen el código de operación más pequeño. Luego se asignarán códigos a las de 3 operandos, y finalmente se verán cuántas combinaciones quedan libres para instrucciones sin operandos.

- Las instrucciones de 1 y de 2 operandos tienen ambas códigos de operación de 8 bits, por lo que pueden estudiarse simultáneamente. Con 8 bits tenemos $2^8 = 256$ combinaciones posibles. Dado que hay que colocar 100+153=253 instrucciones, nos sobran 3 combinaciones. Reservamos entonces las combinaciones `0xFD`, `0xFE` y `0xFF` para extender el código.

Universidad de Valladolid
Departamento de Informática
Escuela Universitaria Politécnica c/Francisco Mendizábal,1 47014 Valladolid Tel.:(983) 423501-02 Fax:(983) 423490

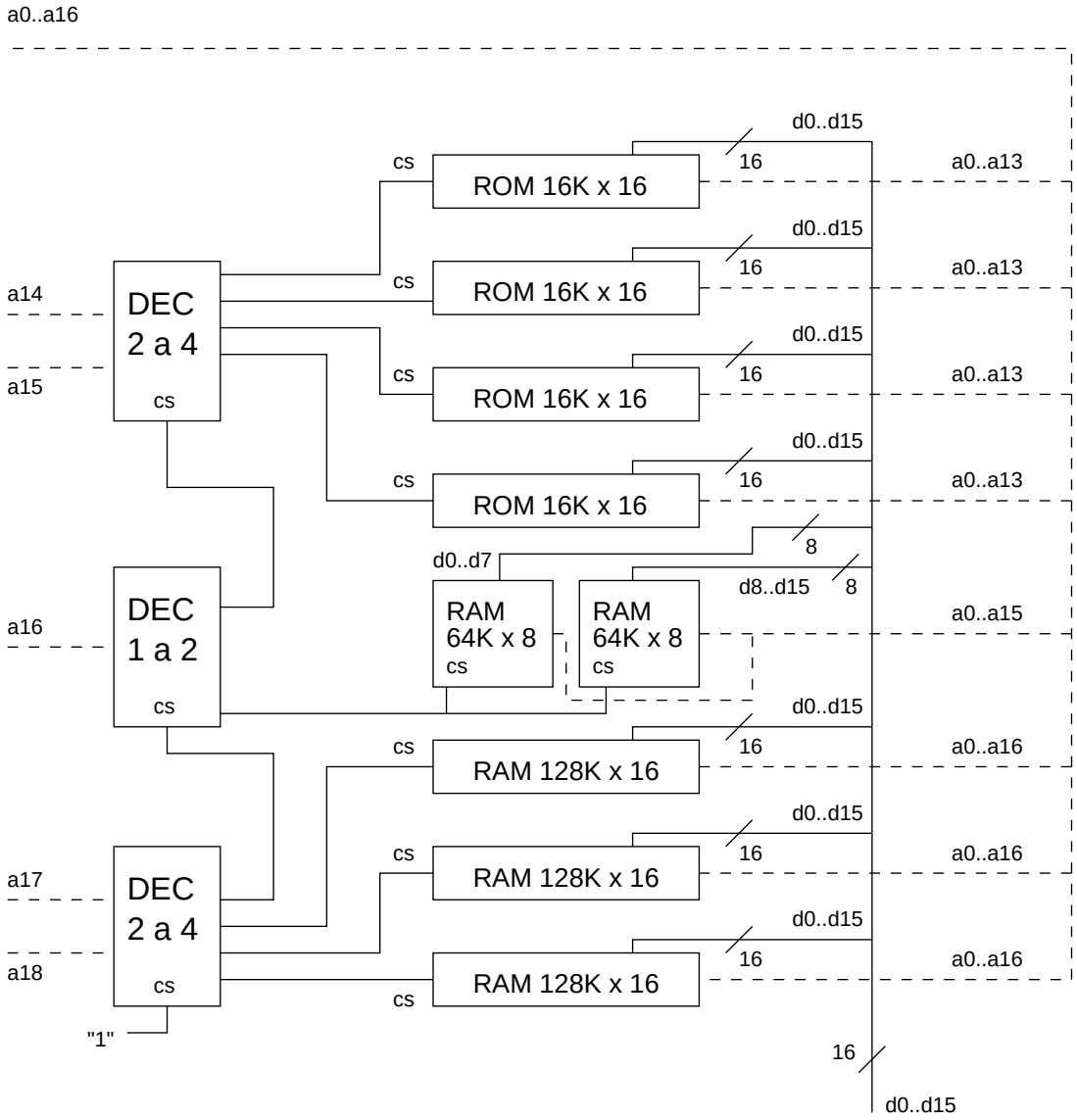

Figura 2: Diagrama de conexión del problema 1.

Universidad de Valladolid
Departamento de Informática
Escuela Universitaria Politécnica c/Francisco Mendizábal,1 47014 Valladolid Tel.:(983) 423501-02 Fax:(983) 423490

- Las instrucciones de 3 operandos tienen 16 bits de código de operación, lo que en principio permitiría codificar 2^{16} instrucciones distintas. Sin embargo, de los 16 bits del código de operación los 8 primeros codifican instrucciones de 1 y de 2 operandos, a excepción de las combinaciones `0xFD`, `0xFE` y `0xFF` ya mencionadas. Por lo tanto, esto nos da un total de $3.2^8 = 768$ instrucciones de tres operandos posibles. Como sólo se necesitan combinaciones para 300 instrucciones de tres operandos, nos sobran $768 - 300 = 468$ combinaciones de los 16 primeros bits.
- Las instrucciones de 0 operandos tienen 32 bits de código de operación. Sin embargo, no pueden utilizarse todos los bits libremente. En concreto, sólo podrá disponerse libremente de los últimos 16 bits, ya que los 16 primeros intervienen en la codificación de las instrucciones de 1, 2 y 3 operandos y por lo tanto sólo podrán usarse las combinaciones que estén libres. Como se ha visto arriba, tras haber colocado las instrucciones de 1, 2 y 3 operandos nos han sobrado 468 combinaciones de los 16 primeros bits. Por lo tanto, el número máximo de instrucciones sin operandos que puede colocarse es de

$$468 * 2^{16} = 30.670.848 \quad (1)$$

Por lo tanto, en las condiciones del problema podrían codificarse más de treinta millones de instrucciones sin operandos. Naturalmente, esto no significa que haya que usar todas las combinaciones.

Cuestión 1

Primera generación de máquinas electrónicas: Se caracterizan por utilizar válvulas de vacio. Aplicaciones: como sistemas de cifrado y descifrado de mensajes enemigos en la segunda guerra mundial (hacia 1940). Ejemplos: ENIGMA (Alemania), COLOSSUS (Reino Unido). Otras aplicaciones: cálculo de tablas de tiro. Ejemplo: ENIAC (Estados Unidos). Otros hechos de importancia: utilización de la arquitectura Von Neumann, en la que las instrucciones se representan en memoria conjuntamente con los datos. Otro ejemplo: IBM 701 año 1953, que introdujo la primera unidad de punto flotante.

Segunda generación: Se destacó por el uso de transistores. El transistor fue inventado por el equipo de John Bardeen, de los laboratorios Bell, en 1948. Nobel diez años mas tarde. Ventajas del transistor frente a las válvulas de vacío: más pequeños (por lo tanto, más rápidos), más baratos, consumen menos energía, mayor vida útil. Ejemplos de sistemas: PDP-1, fabricada por DEC en 1961. La serie continuó con la misma tecnología hasta la PDP-8, mucho mas barata que la primera.

Tercera generación: Se destacó por la aparición de los circuitos integrados, una evolución más rápida y más barata que el uso de transistores. Ejemplos: IBM 360, una línea de 6 máquinas compatibles entre sí. Podía cambiarse de modelo manteniendo el software, dando lugar al concepto de "familia de máquinas". Otro ejemplo: PDP-11, muy popular en las universidades.

Cuarta generación: Se caracteriza por la aparición de las computadoras personales y la utilización de circuitos integrados a gran escala (VLSI). Exponentes de la cuarta generación: IBM PC (ordenador personal), Cray-2 (grandes corporaciones). Además, aparecieron las primeras CPU integradas en una sola pastilla: Intel 8008, 1968. Continúa hasta nuestros días.

Cuestión 2

La familia MIPS está diseñada de modo de acelerar al máximo las invocaciones a las subrutinas. Para ello, utiliza registros para almacenar la dirección de retorno, los parámetros de entrada y el resultado devuelto por la rutina. Los registros usados son los siguientes:

- Dirección de retorno: se almacena en un registro llamado `$ra`.
- Parámetros de entrada a la rutina: se almacenan en los registros `$a0`, `$a1`, `$a2` y `$a3`.
- Resultados: se almacenan en los registros `$v0` y `$v1`.

Si una rutina tiene que invocar a su vez a otra rutina, se presenta el problema de que, para pasarle los parámetros, tiene que hacer uso de los mismos registros que ha utilizado para recibir los parámetros del programa principal, lo que implicaría perder los valores originales allí almacenados. Algo similar sucede con el registro de dirección de retorno y, en algunos casos, con los de devolución de resultados. Para solucionar este problema, lo que la rutina tiene que hacer es volcar el contenido de todos esos registros a la pila, antes de llamar a la segunda rutina. A partir de ese momento podrá utilizar dichos registros para la comunicación con la segunda rutina. Cuando éste haya acabado y haya devuelto los resultados, la rutina recuperará de la pila los parámetros que había guardado previamente y continuará con la ejecución de la tarea correspondiente.

Cuestión 3

El número a representar es $0,0001.10^{-126}$, es decir, 1.10^{-130}. Este número es demasiado pequeño para ser representado, ya que el número más pequeño representable (en formato no normalizado) tiene un exponente de 2^{-149}, que es aproximadamente $1,4.10^{-45}$. En consecuencia, al intentar representar este número se produce una situación de subdesbordamiento, por lo que el número se representa como un cero (con signo positivo). Por lo tanto, la respuesta es `0x00000000`.

Cuestión 4

En notación IEEE 754 de simple precisión se destinan 23 bits al campo de mantisa, 8 al campo de exponente y 1 al de signo de la mantisa. Los números representables son los siguientes:

- Los números en formato normalizado. En este formato, el campo de exponente tiene un valor entre 1 y 254, ya que los

Universidad de Valladolid
Departamento de Informática
Escuela Universitaria Politécnica c/Francisco Mendizábal,1 47014 Valladolid Tel.:(983) 423501-02 Fax:(983) 423490

valores 0 y 255 representan condiciones especiales. Esto hace un total de 254 exponentes distintos posibles. Respecto del campo de mantisa, tiene 23 bits y representa un número de la forma $1, m$ donde m es el campo de mantisa. Como dicho campo tiene 23 bits, esto hace un total de 2^{23} combinaciones posibles para la mantisa. Respecto del signo, puede ser positivo o negativo. Por todo esto, el número total de números reales distintos representables en formato normalizado es

$$X_n = 254.2^{23}.2 = 254.2^{24} \quad (2)$$

- Los números en formato no normalizado. En este caso, el campo de exponente vale cero, y el campo de mantisa representa un número de la forma $0, m$. Como la mantisa tiene 23 bits, esto hace un total de 2^{23} combinaciones. De nuevo tenemos dos signos posibles para la mantisa. El total de combinaciones no normalizadas, es decir, el total de números no normalizados diferentes representables será entonces

$$X_d = 2^{23}.2 = 2^{24} \quad (3)$$

- En tercer lugar, tenemos el caso de las combinaciones que representan al cero. Estas combinaciones son números con mantisa igual a cero y exponente igual a cero, por lo que ya han sido incluidas entre los números no normalizados.
- Finalmente, tenemos el caso del infinito y de la indeterminación, que utilizan el valor 255 en el campo de exponente, y que como se indica en el enunciado no debe considerarse.

De lo expuesto más arriba se deduce que la cantidad de números reales representables será igual a

$$X_n + X_d = 254.2^{24} + 2^{24} = 255.2^{24} \quad (4)$$

Esto hace un total de $255.2^{24} = 4.278.190.080$ números reales representables. No todos ellos son distintos: concretamente, el cero tiene dos representaciones. En consecuencia, el total de números reales *distintos* será la cifra anterior menos 1: es decir, $4.278.190.079$.

Cabe destacar que esta cifra se acerca bastante a 2^{32}, que es el número total de combinaciones posibles con 32 bits, por lo que puede verse que la eficiencia en la codificación es bastante alta.

Cuestión 5

Existen diferentes definiciones posibles para el concepto de ordenador. Un ordenador podría definirse como un sistema capaz de realizar operaciones aritméticas y lógicas con un conjunto de datos de entrada, produciendo un conjunto de datos de salida, y que está compuesto de una unidad central de proceso (CPU), un sistema de memoria y un conjunto de sistemas de entrada y salida, que constituye su interfaz con el exterior. Lo que diferencia a un ordenador de una calculadora es que un ordenador es capaz de almacenar y ejecutar diferentes *programas* que se almacenan en memoria conjuntamente con los datos (arquitectura Von Neumann), mientras que una calculadora es capaz de realizar operaciones aritméticas y lógicas pero siempre bajo la supervisión del usuario. Si una calculadora puede ejecutar programas, estamos hablando de una "calculadora programable" o, lo que es lo mismo, de un ordenador especializado.