El repertorio de instrucciones

1. Un repertorio de instrucciones debe cumplir ciertas exigencias para ser considerado lenguaje de programación. A saber, debe ofrecer soporte para:

   - ejecutar saltos condicionales
   - modificar porciones de memoria

   A la luz de estas exigencias, proponga un conjunto mínimo de operaciones para un repertorio de instrucciones que lo constituya como lenguaje de programación.

2. Enumere qué información de estado es intrínseca al dato. ¿Qué operador *hardware* detecta cada una?

3. ¿Qué información de estado depende del dato y de la operación que lo generó?

4. Sea un procesador que no tiene registro de estado. ¿Cómo se puede detectar un posible acarreo en suma? ¿Y en resta?

5. Según establecieron Böhm y Jacopini en el teorema del programa estructurado (1966) toda tarea computable puede ser escrita combinando estructuras del tipo `if-then-else`, `switch-case`, `for` y `while`. Proponga una implementación de cada una de las cuatro usando instrucciones del lenguaje ensamblador.

6. Utilizar la pila para dar soporte *hardware* a la recursividad de las subrutinas provoca una sobrecarga computacional tanto en número de instrucciones ejecutadas como en tiempo de acceso a memoria. Identifique qué instrucciones se ejecutan en la invocación, retorno y limpieza de una subrutina. Evalúe el tiempo empleado en todos los accesos a la pila suponiendo que cada acceso consume un tiempo igual a $12t_{reg}$ siendo t_{reg} el tiempo de acceso a un registro de la ruta de datos.

7. Sea un computador de instrucción única que ejecuta la operación `restar_y_saltar_si_negativo` (`rsn`) tomando como operandos tres posiciones de memoria `a`, `b` y `c`. La funcionalidad se describe a continuación:

   ```
   rsn a,b,c     mem[a] = mem[a] - mem[b]
                 si (mem[a] < 0) entonces ir a c
   ```

   Como vemos, la instrucción resta el valor contenido en la posición `b` del valor contenido en la posición `a` salvando el resultado en la posición `a`. Si el resultado es menor que 0, se ejecuta la instrucción contenida en la posición `c` y en caso contrario la siguiente.

   Al objeto de cumplir con la funcionalidad de saltar si la diferencia es negativa, los valores contenidos en la memoria se interpretan como enteros codificados en complemento a dos.

   Asumimos que disponemos de una posición de memoria de sólo lectura llamada `uno` que contiene el valor 1.

   Sabiendo todo esto, escriba las secuencias de código que realicen las siguientes operaciones e identifique cada una con un nombre para poderlas usar como macros. Se pueden usar las posiciones de memoria temporales o auxiliares que se desee.

   a) Escriba un 0 en la posición de memoria `destino`.
   b) Copie el contenido de la posición de memoria `fuente` en la posición de memoria `destino`.
   c) Copie en la posición de memoria `neg-fuente` el negativo del valor contenido en `fuente`.
   d) Copie en la posición de memoria `not-fuente` el complemento a uno del valor contenido en `fuente`.

e) Detecte el signo y el cero del operando `fuente`. Si es negativo cargue -1 en el operando de memoria que hace de bandera de signo `sf-fuente` y 0 en caso contrario. Si el operando es cero cargue -1 en el operando que hace de bandera de cero `zf-fuente` y 0 en caso contrario.

f) Escriba una secuencia de código que implemente un salto incondicional a la posición de memoria `#destino`.

g) Escriba una secuencia de código que salte a `#destino` si la bandera de cero `zf-fuente` está activada. La bandera no se puede destruir.

h) Realice el incremento (suma de la unidad) al valor contenido en el operando `fuente`. Asumimos que el operando no tiene signo y no escribimos banderas de estado.

i) Realice el decremento (resta de la unidad) al valor contenido en el operando `fuente`. Asumimos que el operando no tiene signo y no escribimos banderas de estado.

j) Realice la suma de los valores contenidos en las posiciones de memoria `sumando-a` y `sumando-b` salvando el resultado en la primera de ellas. Asumimos que los operandos no tienen signo y no escribimos banderas de estado.

k) Realice la resta del valor contenido en la posición de memoria `sustraendo` del de la posición `minuendo` salvando el resultado en la segunda de ellas. Asumimos que los operandos no tienen signo y no escribimos banderas de estado.

l) Escriba la secuencia de código que realiza el producto de un operando llamado `multiplicando` por otro operando llamado `multiplicador` salvando el resultado en un tercer operando llamado `producto`.

8. Contamos con máquinas de diferentes arquitecturas que podemos agrupar, estudiando sus juegos de instrucciones, de la siguiente forma:

	ARQUITECTURAS				
Modo de ejecución	0 direcciones pila	1 dirección acumulador	2 direcciones registro-registro	3 direcciones memoria-memoria	3 direcciones registro-registro

Juego de instrucciones	`push M`
	`pop M`
	`add`
	`sub`
	`mul`
	`div`
		`load M`
		`store M`
		`add M`
		`sub M`
		`mul M`
		`div M`
			`load X, M`
			`store M, X`
			`move X, Y`
			`add X, Y`
			`sub X, Y`
			`mul X, Y`
			`div X, Y`
				`add M1, M2, M3`
				`sub M1, M2, M3`
				`mul M1, M2, M3`
				`div M1, M2, M3`
					`load X, M`
					`store M, X`
					`move X, Y`
					`add X, Y, Z`
					`sub X, Y, Z`
					`mul X, Y, Z`
					`div X, Y, Z`

Los operandos designados con `M` son posiciones de memoria mientras que los designados con `X`, `Y` o `Z` se refieren a registros del procesador.

La máquina de 0 direcciones tiene un modo de ejecución a pila de forma que todas las operaciones se realizan en la cima de la pila. Las operaciones de proceso toman sus operandos de la cima de la pila y los eliminan sustituyéndolos por el resultado. Las operaciones de transferencia (`push` y `pop`) son las únicas que realizan accesos a memoria.

La máquina de 1 dirección tiene un modo de ejecución basado en acumulador. Las operaciones de proceso se realizan con un operando de memoria y otro implícito que es siempre el acumulador. El juego incluye dos instrucciones de transferencia para cargar el registro acumulador (`load`) o para almacenar su contenido en memoria (`store`).

La máquina de 2 direcciones realiza las operaciones de proceso entre dos operandos residentes en alguno de los 16 registros con que cuenta. Las instrucciones de transferencia son las encargadas de mover información entre los registros y memoria (`load`, `store`) o entre los propios registros (`move`).

Las máquinas de 3 direcciones especifican tres operandos por instrucción. Se proponen en la tabla dos casos extremos: uno de ellos trabaja en modo de ejecución memoria-memoria mientras que el otro es registro-registro (también llamado de carga/almacenamiento).

Sabiendo todo esto, escribir los programas correspondientes a cada juego de instrucciones propuesto de forma que realicen el siguiente cálculo:

$$A = \frac{(B + C) \cdot D}{E - F \cdot G - H \cdot I}$$

Partir de la situación en la que todas las variables están en memoria.

9. Mida la eficiencia de memoria de las diferentes arquitecturas propuestas en el problema anterior. Se harán las siguientes suposiciones sobre los repertorios de instrucciones:

   - los códigos de operación son de 1 byte
   - el direccionamiento de los operandos de memoria ocupa 2 bytes
   - los operandos son de 2 bytes
   - los registros se especifican mediante campos de 4 bits (16 registros)

   Calcule para cada programa dado en la solución del problema anterior el número de bytes que ocupa y el número de bytes que se transfieren a través del bus de datos. Determinar cual es la arquitectura más eficiente si solo se tiene en cuenta el tamaño del código ejecutable. Determinar la eficiencia si se evalúa desde el punto de vista del ancho de banda total de memoria que se necesita, es decir, la suma de bytes de código más datos que se han de mover.

10. Los accesos a datos en memoria se pueden implementar mediante direccionamientos directos, relativos a registro o indirectos. Describa cada uno de ellos y discuta sus ventajas e inconvenientes.

11. El modo de direccionamiento indexado con autoincremento o autodecremento es similar al relativo a registro índice pero con la particularidad de que incrementa o decrementa el índice automaticamente antes o después de calcular la dirección efectiva de memoria. Discuta las ventajas e inconvenientes que presenta este modo.

12. En muchos casos, el modo de direccionamiento indexado con autoincremento o autodecremento suma o resta un valor diferente en función del objeto referenciado. Explique por qué se hace esto así y hasta qué punto es interesante.

13. Sabemos que el modo de direccionamiento absoluto a memoria no es útil en máquinas con un mapa de memoria muy grande ya que incluir los punteros en el formato de las instrucciones las hace desmasiado largas. Por ello se creó el direccionamiento paginado a memoria que divide la memoria en páginas de tamaño manejable referenciadas por un puntero *base* y luego se direcciona la memoria mediante un puntero pequeño a la página llamado *desplazamiento*. Discuta las ventajas e inconvenientes de este modo de direccionamiento.

14. Sea una máquina de acumulador (`Acc`) de la que se quieren diseñar los formatos de representación de sus instrucciones teniendo en cuenta las siguientes características del computador:

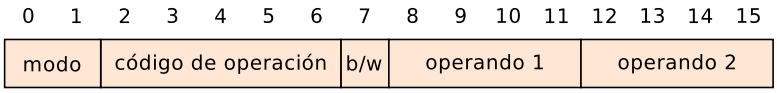
    - el tamaño de palabra es de 16 bits;
    - el modelo de ejecución es registro-registro;
    - la memoria principal es de 2 Kpalabras;
    - tiene 32 registros, entre los cuales se encuentra el acumulador (`Acc`), el `PC` y el `SP`;
    - los modos de direccionamiento que puede tener el procesador son: inmediato, directo a registro y a memoria, relativo a registro base y relativo a `PC`;
    - las operaciones que ejecuta son:
      - instrucciones de una dirección:
        * `STA`: almacena el contenido del `Acc` en un registro o en una dirección de memoria.
        * `LDA`: carga en el Acc un inmediato, el contenido de una posición de memoria o el de un registro.

    - `Bcc`: salto condicional (las posibles condiciones son `Z`, `NZ`, `C`, `NC` y el único modo de direccionamiento para esta instrucción es relativo a `PC`).
    - `BR`, `CALL`, `PUSH`, `POP`, `ADD`, `SUB`, `MUL`, `AND`, `OR`, `XOR`, `CMP`
  - instrucciones de cero direcciones:
    - `RET`, `INC`, `DEC`, `NEG`, `NOT`, `SHL`, `SHR`, `ROL`, `ROR`, `HALT`, `WAIT`

  Sabiendo todo esto, diseñe el/los formatos de representación de instrucciones, indicando los modos de direccionamiento que puede admitir cada instrucción (o grupo de instrucciones), y minimizando el espacio de representación.

15. Sea un repertorio de instrucciones cuyo formato tiene una longitud fija de 16 bits. Las direcciones (especificación de operando) se realizan sobre 6 bits. Cuenta con K instrucciones de 2 direcciones, L instrucciones de 1 dirección y M instrucciones de 0 direcciones.
    El código de operación es de longitud variable. El campo *opcode* tiene un campo de extensión y el código de operación extendido que lleva asociado también cuenta con un campo de extensión anidado con su correspondiente código de operación extendido. El grupo de K instrucciones se codifica sobre el *opcode* sin campos de extensión, el grupo de L instrucciones deriva del campo de extensión de primer nivel y el grupo de M instrucciones del campo de extensión anidado.
    Determine los valores máximos de K, L y M, y el número máximo de instrucciones que puede permitirse este formato,

16. El procesador Zilog Z8001, de 16 bits, tiene el siguiente formato de instrucción:



    El campo modo especifica los modos de direccionamiento utilizados en los campos operando 1 y operando 2. El campo código de operación especifica la operación a realizar. El campo b/w determina si los operandos son de tamaño byte o palabra. Los campos operando 1 y operando 2 señalan uno de los 16 registros del procesador. Cuando el campo operando 2 contiene todo ceros, en lugar de identificar el registro 0, significa que cada uno de los códigos del código de operación tienen un significado distinto. ¿Cuántas operaciones diferentes es capaz de ejecutar el Z8001? Sugiera una manera alternativa de disponer de más códigos de operación indicando los compromisos a adoptar.

17. La codificación de un repertorio de instrucciones se ha diseñado con el objetivo de minimizar el espacio de representación. De esta manera, se han especificado dos formatos para los códigos de operación: uno "regular" y otro "extendido". El formato "regular" codifica las operaciones sobre un campo de 6 bits reservando 8 combinaciones como campos de extensión. El formato "extendido" asocia cada uno de esos campos de extensión con un código de operación extendido de 3 bits.

    Determine la cantidad máxima de operaciones codificables en el formato "regular" y en el "extendido". ¿Qué cantidad de bits se necesitarían para codificar todas esas operaciones si el formato fuera solamente "regular"? Calcule el tamaño medio de codificación de este diseño si el formato "regular" se utiliza en el 80% de los casos. Y si todas las operaciones se utilizaran por igual, ¿cuál sería su tamaño medio? Finalmente, determine el mínimo porcentaje de uso del formato "regular" que hace bueno el diseño propuesto.

18. Sea un repertorio de instrucciones con 92 operaciones distintas. Una serie de pruebas realizadas con programas de *test* muestran que solamente 12 operaciones son responsables del 80% de las instrucciones procesadas. Sabiendo esto, se pretende minimizar el espacio de representación utilizando un formato regular y otro extendido. Determine el tamaño del código de operación del formato regular y el número de los campos de extensión y el tamaño del código de operación adicional en el formato extendido. Calcule el tamaño promedio efectivo del código de operación para los programas de prueba.

19. Las máquinas registro-registro realizan las operaciones de proceso de acuerdo al patrón del siguiente ejemplo:

```
load r1, [rb]       ;r1 <- [rb]
add r2, r2, r1      ;r2 <- r2 + r1
```

Algunos proponen crear un nueva instrucción con modo de direccionamiento registro-memoria de manera que la secuencia anterior se sustituya por esta otra:

```
add r2, [rb]        ;r2 <- r2 + [rb]
```

¿Qué ventajas e inconvenientes presenta la propuesta?

La distribución del uso de instrucciones para un programa es la siguiente:

tipo de instrucción	**porcentaje**
operaciones de transferencia	36%
operaciones de bifurcación	18%
operaciones de proceso	46%

De las operaciones de transferencia 2/3 son cargas y el resto almacenamientos.

Determine qué porcentaje de las instrucciones de carga debe eliminarse para que el modelo alternativo propuesto más arriba tenga al menos el mismo rendimiento que la máquina registro-registro típica. Suponemos que el ciclo de reloj se incrementa en un 10% y que el CPI permanece igual.