Nº Orden	Apellido y nombre	L.U.	Cantidad de hojas

Organización del Computador 2
Primer parcial – 15/05/2012

1 (40)	2 (40)	3 (20)	Total

Normas generales

- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Entregue esta hoja junto al examen, la misma **no** se incluye en la cantidad total de hojas entregadas.
- Está permitido tener los manuales y los apuntes con las listas de instrucciones en el examen. Está prohibido compartir manuales o apuntes entre alumnos durante el examen.
- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido y LU.
- La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los parciales tienen tres notas: I (Insuficiente): 0 a 59 pts, A- (Aprobado condicional): 60 a 64 pts y A (Aprobado): 65 a 100 pts. No se puede aprobar con A- ambos parciales. Los recuperatorios tienen dos notas: I: 0 a 64 pts y A: 65 a 100 pts.

Ej. 1. (40 puntos)

Sea una matriz de enteros sin signo de 16 bits de tamaño $n \times m$; donde n y m son múltiplos de 8. Se desea construir una función que obtenga la sumatoria de todos los valores dentro de la matriz que sean multiplo de 4. Esta sumatoria entra en 29 bits.

```
int losMultiplosDe4(unsigned int n, unsigned short* matriz, unsigned int m)
```

1. (30 puntos) Programar en ASM usando instrucciones SIMD el código de la función pedida.
2. (10 puntos) Explicar por qué su solución respeta la condición impuesta sobre las dimensiones de la matriz.

Nota: Un número es múltiplo de 4 $\iff$ los dos últimos dígitos de su representación binaria son 0.

Ej. 2. (40 puntos)

Dada una estructura de tabla de Hash se busca hacer una función que agregue un elemento a la misma. Para calcular la entrada en la tabla se provee una función que permite obtener el hash, con la siguiente aridad:

```
int evalHash(void*)
```

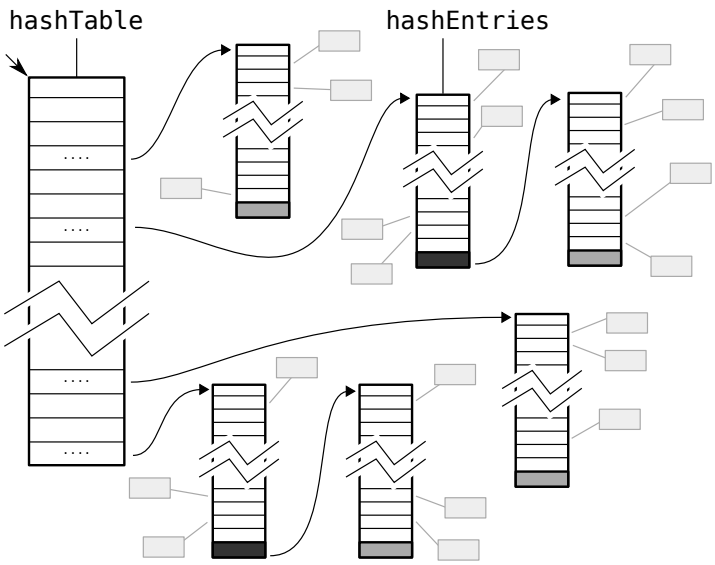
El valor de retorno del Hash es un número entre 0 y 1023.

La estructura de tabla de Hash es la siguiente:

```
typedef struct s_hashTable {
  hashEntries* hashEntries[1024];
}__attribute__((packed)) hashTable;

typedef struct s_hashEntries {
  void *entries[256];
  void *others_entries;
}__attribute__((packed)) hashEntries;
```

La función agregar (`void agregar(hashTable* ht, void* clave, void* entrada)`) se encarga de calcular la función de hash para la clave dada y obtener el índice correspondiente. Luego debe recorrer la estructura de `hashEntries`, hasta encontrar un lugar vacío donde colocar la nueva entrada. Se considera que el lugar es vacío si el mismo tiene un puntero a NULL. En el caso que no se encuentre ningún lugar vacío, o que no exista ninguna estructura de `hashEntries`; se debe crear y completar una nueva.



1. (10 puntos) Escribir el pseudocódigo de la función pedida.
2. (30 puntos) Programar en ASM el código de la función anterior.

Ej. 3. (20 puntos)

Sea el siguiente código C:

```
const char * juan = "juan";
const double trespuntoocho  = 3.8;

int f(char y, int x) {
  char* formato = "_%i\n";
  formato[0] = y;
  printf(formato, f1(1, f2(trespuntoocho, x), f3(f4(juan))));
  return 35;
}
```

donde las funciones tienen la aridad:

- `int f1(unsigned int, double, int);`
- `double f2(double, int);`
- `int f3(double);`
- `double f4(char*);`

1. (4 puntos) Identificar si la variable `formato` es local a la función, o global a todo el código. ¿Por qué?
2. (16 puntos) Programar en ASM el código anterior.

Nota: No se puede asumir que los registros de multimedia se conservan durante los llamados a las funciones.