

**EXAMEN DE PRÁCTICAS**  
**Convocatoria extraordinaria**  
**ARQUITECTURA DE SISTEMAS AUDIOVISUALES II**  
**4º SAM, URJC**

*Fuenlabrada (Examen remoto), 3 de Julio de 2020*

**AVISO:** Asegúrate que tus programas **cumplen** con los siguientes criterios. Si no se cumple alguno de ellos la **nota máxima** de tu examen será de **2 puntos**

- **Cumplimiento de especificaciones.** Se deben cumplir las especificaciones indicadas en el enunciado: nombres de funciones, nombres de archivos, funcionalidad, etc. Compruébalo antes de entregar el examen
- **Respetar el convenio.** Resuelve las preguntas **sin violar** el convenio del uso de registros (ABI del RISC-V)
- **Sin errores en tiempo de ejecución** (Runtime errors). Tus programas no deben generar excepciones al ejecutarse
- **Sin errores al ensamblar.** Los ficheros entregados NO deben dar errores al ensamblarlos. Si una función la has dejado a medio hacer, asegúrate que al menos se ensambla sin errores

Nuestro jefe de proyecto necesita disponer en el RISC-V de la capacidad para calcular **potencias**, a partir de una **base** y un **exponente** enteros. Por requisitos del proyecto, se debe usar un **RISC-V básico**, que **NO** incluye las instrucciones de multiplicación. Por tanto, los cálculos se deben hacer utilizando sólo **instrucciones de suma**

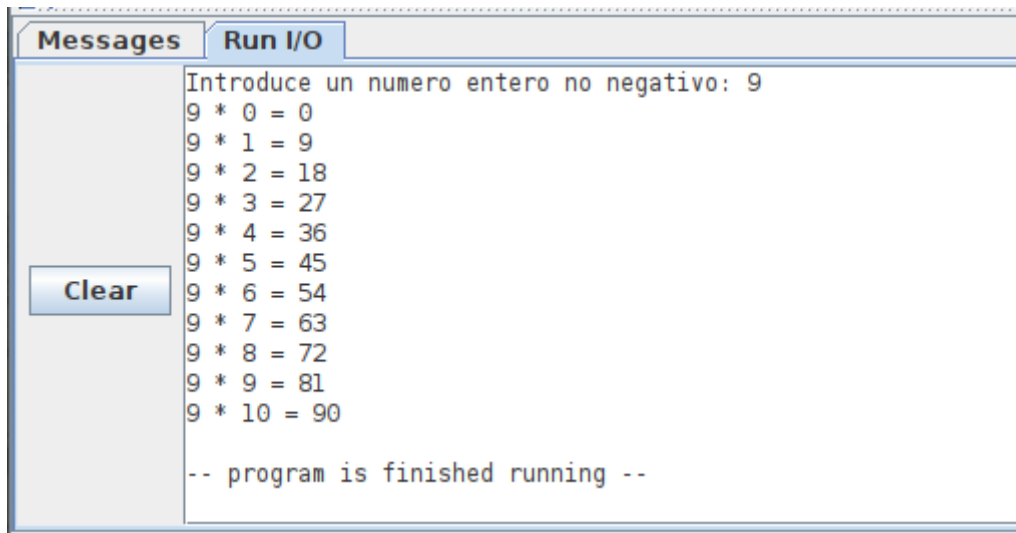
En esta fase del proyecto no se está interesado en la optimización de velocidad, sino simplemente en lograr la funcionalidad. En fases posteriores se realizarán optimizaciones

Para lograr este objetivo, nos ha dado las **especificaciones** de las siguientes **funciones**, que debemos implementar en lenguaje ensamblador:

- *int mult(a, b):* Multiplicación de dos números enteros no negativos ( $a \geq 0$ ,  $b \geq 0$ ). Esta función recibe como **argumentos de entrada** los números enteros  $a$  y  $b$ , de 32 bits y devuelve como **argumento de salida** su multiplicación. Se utilizará un **algoritmo iterativo** que sume un operando consigo mismo tantas veces como indique el otro operando. **Sólo** se pueden utilizar **instrucciones de suma** del Risc-V para los cálculos. Centrar el esfuerzo en que el resultado de la multiplicación sea correcto (sin realizar optimizaciones de velocidad)
- *int exp(a, n):* Cálculo de potencias. La función recibe como **argumentos de entrada** los números enteros  $a$  y  $n$ , donde  $a$  es la **base** ( $a \geq 0$ ) y  $n$  el **exponente** ( $n > 0$ ). Devuelve como argumento de salida el resultado de elevar  $a$  al exponente  $n$ , es decir, multiplicar  $a$  por sí misma tantas veces como indique el exponente. Se debe **llamar** a la función **mult()** para realizar las multiplicaciones necesarias

Para **probar** las funciones se realizarán los siguientes **programas de prueba** (programas principales)

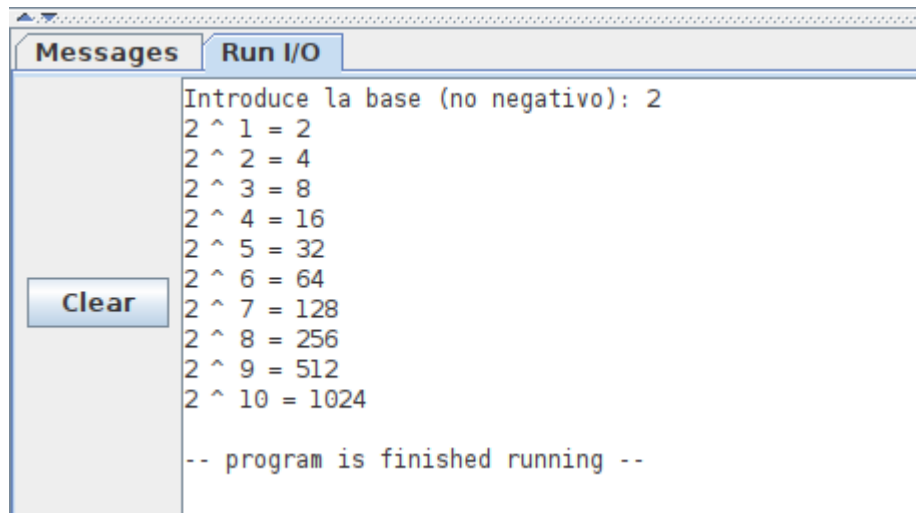
- *test\_mult.s*: **Programa** para probar la función de multiplicación. Se pide al usuario que introduzca un **número entero** y que imprima en la consola la tabla de multiplicación, mostrando una operación en cada línea. Así, por ejemplo, si introducimos el número 9, en la consola obtendremos esto:



```
Messages Run I/O
Introduce un numero entero no negativo: 9
9 * 0 = 0
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90

-- program is finished running --
```

- *test\_exp.s*: **Programa** para probar la función de cálculo de potencias. Se pide al usuario que introduzca la base e imprime en la consola los cálculos para el exponente desde 1 hasta 10, una operación en cada línea. Por ejemplo, si el usuario introduce la base 2, la salida será la siguiente:



```
Messages Run I/O
Introduce la base (no negativo): 2
2 ^ 1 = 2
2 ^ 2 = 4
2 ^ 3 = 8
2 ^ 4 = 16
2 ^ 5 = 32
2 ^ 6 = 64
2 ^ 7 = 128
2 ^ 8 = 256
2 ^ 9 = 512
2 ^ 10 = 1024

-- program is finished running --
```

**Se pide:**

1. Implementar la función `mult()` en el fichero **mult.s**. (2 puntos)
2. Implementar el programa de prueba **test\_mult.s** (3 puntos)
3. Implementar la función `exp()` en el fichero **exp.s** (3 puntos)
4. Implementar el programa de prueba **test\_exp.s** (2 puntos)

Asegúrate de que tus programas sólo funcionan con esos ficheros (no podrás incorporar ningún fichero adicional). Define todas las constantes que necesites dentro de esos ficheros