# Proyecto 1: Emulador MIPS en ensamblador x_86

Profesor: Ernesto Rivera Alvarado

## I. Introducción

La virtualización de arquitecturas legado es un mecanismo mediante el cual se puede perpetuar el software creado para sistemas computacionales que dejaron de existir. Un ejemplo de ello son los conocidos como **emuladores** de consolas de videojuegos tales como NES, SNES, Play Station, Play Station 2, Play Station 3, Atari, Wii, Game Cube, Game Boy, Sega Genesis, así como los compilados de juegos y la consola virtual de Nintendo, entre muchos otros. Debido a las restricciones en rendimiento y de temporización que implica el ejecutar código máquina de una arquitectura distinta al procesador donde efectivamente se ejecuta, se suelen utilizar lenguajes de mediano o bajo nivel para esta tarea, debido a que permite obtener alto rendimiento y realizar manipulaciones de bajo nivel en la memoria. En este proyecto implementará un emulador de la arquitectura MIPS en lenguaje ensamblador x_86 sobre Linux. Para ello, el estudiante hará uso del ensamblador NASM para leer un archivo de texto que contiene el código ROM MIPS a ejecutar, y utilizará la consola de texto Bash como ambiente gráfico para desplegar resultados.

## II. Configuración del sistema y código base

- Para desarrollar el proyecto, es necesario que el estudiante instale en su computadora un sistema operativo Linux de su preferencia (no se admite el uso de máquina virtual).
- Posterior a esto, es necesaria la instalación del ensamblador NASM.
- El estudiante utilizar como base el código proporcionado por el profesor en el Tec-Digital para el desarrollo del ambiente gráfico..
- Todo el código debe realizarse en ensamblador x_86 sobre Linux, no se permite invocar rutinas en otros lenguajes de programación desde el programa desarrollado por el usuario.

## III. Descripción

Los requerimientos del emulador se describen a continuación:

- Debe realizar un emulador capaz de correr cualquier código máquina MIPS en formato hex generado a partir del simulador MARS. Las entradas al emulador es un archivo de texto con el .text y .data del ROM en MIPS. Dichos archivos de texto pueden estar en formato binario o hexadecimal.
- Dicho emulador tiene que tener implementado las funcionalidades básicas de MARS tal como despliegue de video, captura de datos de entrada y los diferentes `syscall` que usted utilizó en su proyecto programado en MIPS.
- Debe ser capaz de ejecutar el pong.asm sin problema, así como otros códigos que el profesor podrá proveer en el día de la revisión, tal como lo ejecutaría el simulador MARS.
- Debe simular a nivel de software todo el hardware que se visualiza en la microarquitectura MIPS, tal como el PC, banco de registros, stack, memoria de programa, memoria de datos, etc.
- No debe presentar *bugs*, ni *glitches* ni comportamientos indeseados.
- Cada vez que el programa ejecuta y decodifica una instrucción, esta debe ser guardada en un archivo de texto con todos sus detalles, incluyendo la dirección del PC en la cual se encuentra.
- Los archivos ROM de MIPS .text y .data deberán ingresarse por consola de comandos al ejecutar el emulador. Entiéndase, la ejecución del emulador sería de la siguiente manera: `./emulador .text .data`. Si el estudiante prefiere, puede consolidar el .text y .data en un solo archivo.

Considere que las características mencionadas anteriormente deben funcionar de manera correcta para que se consideren en el proyecto.

## IV. Sobre la revisión y completitud del proyecto

Es de suma importancia que el estudiante incluya dentro de su proyecto todas las características mencionadas en la descripción, ya que este será un aspecto que se evaluará fuertemente en la revisión del mismo. Por cada característica que el estudiante no incluya, se rebajará de la nota final $3^k$ puntos hasta un máximo de 40, donde $k$ corresponde a la cantidad de características que usted no incluyó, esto aplica para $k > 1$. Adicional a esto, las funcionalidades no incluidas, o que no funcionan correctamente en el proyecto se rebajarán adicional al rubro de completitud mencionado anteriormente.

La evaluación de este proyecto está sujeta a la presentación oral del mismo, en la que el estudiante debe mostrar un dominio completo del trabajo realizado a nivel de código. Por cada falla a la hora de explicar el funcionamiento del código desarrollado, se rebajarán 5 puntos del total.

## V. Requisitos indispensables

La ausencia de uno solo de los siguientes requisitos vuelve al proyecto "no revisable" y recibe un 0 de calificación inmediata:

- Se sigue a cabalidad lo solicitado en la Sección II del presente documento.
- No debe presentar "*Segmentation Fault*" o "Violación de segmento" por ninguna razón.

## VI. Desarrollo del proyecto

El proyecto está pensado para desarrollarse individualmente, sin embargo, los estudiantes que deseen reforzar la habilidad de trabajo en equipo pueden entregar el proyecto en parejas. El profesor les hace la aclaración de que el trabajo con un compañero conlleva dificultades de coordinación, división de trabajo y sobre todo de "pegar o juntar ambas partes". En experiencias propias del profesor, se les comenta que en ocasiones el trabajo de juntar, acoplar y corregir partes desarrolladas por diferentes personas conlleva más tiempo y trabajo que la realización individual.

## VII. Consideraciones sobre plagio

El trabajo presentado por los estudiantes (ya sea a nivel individual y de parejas) debe ser de su propia autoría. No se permite utilizar código realizado por un tercero (sin importar la licencia de dicho código) o entre compañeros del mismo curso, y en caso de que esto ocurra, será considerado plagio por lo que se seguirá el proceso correspondiente. Códigos encontrados en "github" de los cuales solo se tomaron "partes" será considerado plagio, el estudiante es el responsable de desarrollar absolutamente todo su código, a excepción del código que se menciona como base en el presente documento.

## VIII. Fecha de entrega

Las demostraciones se harán en la **semana 5** asignados por citas distribuidas aleatoriamente en las dos lecciones de la semana.

La carpeta comprimida `.zip` de su proyecto debe contener los archivos fuentes `.c` y complementarios, además del archivo `Makefile` que genera el ejecutable del código fuente. El nombre de la carpeta comprimida debe de ser los apellidos de integrantes del grupo de trabajo (por ejemplo `rivera-alvarado.zip`) y este será subido al Tec-Digital el día de la revisión a antes de las 6 am.