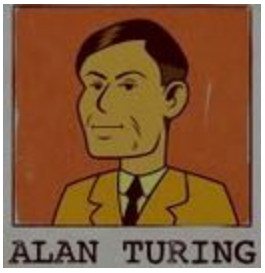
Organización del Computador
TP: Ensamblador ARM

Contexto :
La Inteligencia Artificial es el área de la computación encargada de evaluar e implementar programas que puedan resolver problemas aprendiendo a partir de los datos de entrada y generando nuevas soluciones.

Se dice que un programa utiliza inteligencia artificial si es capaz de pasar el test de Turing (inventado por Alan Turing 1912-1954):

1. Se realiza un experimento en el cual una persona interactúa con el programa sin saber si este es un humano o un software.
2. Si la persona no puede distinguir si el interlocutor es un programa o una persona entonces el programa implementa Inteligencia Artificial.



Hoy en día los programas que intentan romper el test de Turing son los famosos bots. Un ejemplo es el bot de la propaganda del banco Galicia:

En el presente TP vamos a crear una inteligencia artificial! Pero cuidado, los intentos de crear este tipo de tecnología pueden terminar con la dominación del mundo como nos cuenta Terminator, Matrix y este capítulo de una serie animada: link

Consigna

Realizar un programa en assembler que implemente un bot, es decir, un programa que lee los mensajes tipeados por el usuario y devuelve una respuesta de acuerdo al mensaje del usuario. La implementación debe contar con los siguientes puntos:

- El procesamiento del bot debe responder a consultas aritméticas, por ejemplo, sumar un par de números, o restarlos. Los mensajes deben restringirse al lenguaje aceptado por el bot que se describe más adelante en consideraciones generales.

- Extender el procesamiento del bot para que dé una respuesta cordial en caso de error, es decir, cuando la expresión tipeada por el usuario no pertenece al lenguaje aceptado por el bot. Por ejemplo, puede responder: “Lo siento, mis respuestas son limitadas.” como era la respuesta del holograma en la película Yo Robot. De esta manera intentaremos acercarnos al test de Turing!

- Extender el procesamiento del bot para que reconozca la palabra adiós para cerrar el programa y despedirse.

Consideraciones generales

- **Esquema general pseudocódigo**

El esquema general del programa que se provee está inspirado en el ciclo de instrucción (Fetch-Decode-Execute) y es el siguiente:

```
Mientras el usuario no decida salir{
   //etapa Fetch
   Leer el input del usuario

   //etapa Decode
   Procesar el input, distinguir si es una solicitud aritmética o un mensaje normal

   //etapa Execute
   Procesar la respuesta adecuada para el usuario e imprimir por pantalla
}
```

Las funcionalidades solicitadas se deben agregar en forma de llamadas a subrutinas. Comentar todas las instrucciones, aclarando comportamiento y resultado esperado de la instrucción.

Atención: ver las subrutinas y secciones de datos sugeridas más adelante en el punto "Subrutinas y sección de datos". El docente aceptará o sugerirá cambios para que empiece a programar.

- **Lenguaje aceptado por el bot**

Una instrucción aritmética debe tener la siguiente sintaxis:

Operando1 espacio Operación espacio Operando2

Donde
Espacio: Debe existir espacio entre los operandos y la operación.

Operando1 y Operando2: son enteros (número positivo o negativo) en base decimal que pueden tener 1, 2 o más dígitos. El programa debe reconocer los valores enteros para almacenarlos en un registro en Complemento A2 y luego realizar la operación aritmética.
Para el caso de la división los operandos negativos pueden traer problemas, prestar atención.

Operación: Puede ser suma (+), resta (-), multiplicación (*) o división entera (/).

Debe existir espacio entre los operandos y la operación.

Ejemplo de mensaje aceptado: La siguiente expresión suma los valores 12 y -15
12 + -15

Ejemplo de mensaje rechazado: 4+3 La siguiente expresión no pertenece al lenguaje aceptado por el bot (no hay espacios separando las letras)

- **Subrutinas y sección de datos**

Para la parte de datos se sugiere utilizar los siguientes datos:

```
.data
input_usuario:    .asciz "                                                    "
mensaje_error:  .asciz "Lo siento, mis respuestas son limitadas \n"
text_result:        .asciz "##########"
operacion:          .byte 0
num1:               .int 0
num2:               .int 0
resultado:          .int 0
resto:              .int 0
mensaje_despedida:      .asciz "Adios! \n"
```

Para la parte de **subrutinas** se sugiere implementar:

Subrutina	Descripción
leer_input_usuario:	Se ingresa input del usuario
es_cuenta:	Controla la construcción dígito espacio operador espacio dígito Sugerencia: puede devolver las posiciones de los operandos y el operador, para poder trabajar después
obtener_valor:	Obtiene el valor numérico de un operando. Hay que pasarle el puntero a la primera posición del número
resolver_operacion:	Resuelve la operación
division:	Se realiza la división entera y se guarda el resto en memoria Sugerencia: realizar la división con el módulo y luego agregar el signo
imprimir:	Imprime por pantalla un mensaje, los parámetros son el puntero al texto a imprimir y la cantidad de bytes
convertir_texto:	Convierte un número a su representación ascii Tener en cuenta que los números negativos están en complemento a dos
es_salir:	Controla si el usuario escribió una despedida

Entrega de tp:

- Entregar vía moodle el archivo zip conteniendo el código fuente, debidamente comentado y un documento con el un informe detallando el funcionamiento de su programa junto con una descripción de las decisiones de implementación que se hubieran tomado.
- El informe debe estar en formato PDF y debe respetar los siguientes puntos: texto justificado, letra tamaño 12, datos de integrantes y comisión en la carátula.
- **No incluir archivos objeto o ejecutables**.
- Nombrar a su archivo: TP_ARM_com-0X_GX_Apellido_Nombre.zip.
- Resolver en grupos de a tres personas.
- Subir a la entrega correspondiente del moodle el zip con el código fuente y el informe en formato PDF.
- Se dispone de dos semanas para resolver el tp.
- Fecha de entrega: Martes 23 de Junio hasta las 10:00hs. Se habilitará en moodle una sección para que puedan entregar el tp hasta esa fecha y hora.