Macros de una máquina RAM

Implementaremos las macros de la máquina RAM como módulos de Mathematica.

Todas las macros actúan sobre una estructura de tipo lista externa que hemos denominado "memoria" y que simula a la memoria de la máquina.

Por restricciones de implementación consideraremos que la memoria empieza a enumerarse en la dirección '1' en vez de en la dirección '0'.

Las etiquetas de instrucción que aparecen como resultado de algunas macros las consideraremos cadenas de texto y no les declararemos ningún tipo del Mathematica de forma explícita.

Instrucción sucesor

```
In[•]:= suc[i_Integer] := Module[{},
          entrada        módulo
        memoria[[i]] = memoria[[i]] + 1
       ];
```

Instrucción predecesor

```
In[•]:= pre[i_Integer, l_] := Module[{label},
          entrada          módulo
        If[memoria[[i]] > 0, memoria[[i]] = memoria[[i]] - 1, Goto[l]];
          si                                                  ve a
       ];
```

Instrucción cero

```
In[•]:= cer[i_Integer] := Module[{},
          entrada        módulo
        memoria[[i]] := 0
       ];
```

Instrucción asignación

```
In[•]:= asi[c_Integer, i_Integer] := Module[{},
          entrada      entrada      módulo
        memoria[[i]] = c
       ];
```

Instrucción copia

```
In[•]:= cop[j_Integer, i_Integer] := Module[{},
          entrada      entrada      módulo
        memoria[[i]] = memoria[[j]]
       ];
```

Instrucción suma

```
In[•]:= sum[i_Integer, j_Integer, k_Integer] := Module[{},
          entrada   entrada   entrada   módulo
         memoria[[k]] = memoria[[i]] + memoria[[j]]
        ];
```

Instrucción multiplicación

```
In[•]:= mul[i_Integer, j_Integer, k_Integer] := Module[{},
          entrada   entrada   entrada   módulo
         memoria[[k]] = memoria[[i]] * memoria[[j]]
        ];
```

Instrucción división entera

```
In[•]:= div[i_Integer, j_Integer, k_Integer] := Module[{},
          entrada   entrada   entrada   módulo
         If[memoria[[j]] == 0, Return["error"],
         si                    retorna
          memoria[[k]] = IntegerPart[memoria[[i]] / memoria[[j]]]]
                         parte entera
        ];
```

Instrucción comparación menor o igual

```
In[•]:= mei[i_Integer, j_Integer, l1_, l2_] := Module[{},
          entrada   entrada             módulo
         If[memoria[[i]] ≤ memoria[[j]], Goto[l1], Goto[l2]]
         si                              ve a      ve a
        ];
```

Instrucción comparación igual

```
In[•]:= ig[i_Integer, j_Integer, l1_, l2_] := Module[{},
          entrada   entrada             módulo
         If[memoria[[i]] == memoria[[j]], Goto[l1], Goto[l2]]
         si                               ve a      ve a
        ];

In[•]:= memoria = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Out[•]= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

Ignacio Diago Valeta

```
(*Cada función tiene un nombre fx con x ∈ {1,2,3,4} dependiendo del ejercicio
  que estoy resolviendo y siguiendo el orden de numeración del boletín*)

(*Esta función f5 la desarrollé porque para hacer la potencia
 de una raíz necesitaba pensar primero como hacer la potencia,
la dejo por si quieres ver el proceso de desarrollo de mi código,
luego copié y pegué y cambie las etiquetas de nombre*)
```

Los registros 5, 6, 7, 8 y 9 no se usan.

In[ ]:=
```
f5[base_Integer, exponente_Integer] := Module[{}, cop[base, 2];
                                       módulo
  cop[exponente, 3];
  cer[10];
  ig[3, 10, "next", "next2"];
Label["next"]; asi[1, 1]; Goto["end"];
etiqueta                  ve a
Label["next2"]; cer[10]; ig[2, 10, "next3", "next4"];
etiqueta
Label["next3"]; asi[0, 1]; Goto["end"];
etiqueta                   ve a
Label["next4"]; asi[1, 4];
etiqueta
Label["iter"]; pre[3, "end"]; mul[2, 4, 4]; cop[4, 1]; Goto["iter"];
etiqueta                                               ve a
Label["end"];
etiqueta
  Print["El resultado de elevar la base seleccionada
  escribe
     a la raíz del exponente seleccionado es: ", memoria[[1]];]

(*Aquí es donde empiezan los ejercicios propuestos*)
```

Potencia de un número elevado a la raíz cuadrada de otro número:

Los registros 8 y 9 no se usan.

```
In[ ]:= f1[base_Integer, exponente_Integer] := Module[{}, cop[exponente, 2];
                                                módulo
    cop[base, 7];
    cer[3];
    cer[4];
    cer[5];
    cer[6];
    ig[2, 6, "end", "iter"];
  Label["iter"];
  etiqueta
    cop[4, 6];
    suc[3];
    suc[4];
    mul[3, 4, 5];
    mei[5, 2, "iter", "end"];
  Label["end"];
  etiqueta
    cop[7, 2];
    cop[6, 3];
    cer[10];
    ig[3, 10, "next", "next2"];
  Label["next"]; asi[1, 1]; Goto["end2"];
  etiqueta                  ve a
  Label["next2"]; ig[2, 10, "next3", "next4"];
  etiqueta
  Label["next3"]; asi[0, 1]; Goto["end2"];
  etiqueta                   ve a
  Label["next4"]; asi[1, 4];
  etiqueta
  Label["iter2"]; pre[3, "end2"]; mul[2, 4, 4]; cop[4, 1]; Goto["iter2"];
  etiqueta                                                  ve a
  Label["end2"];
  etiqueta
    Print["El resultado de elevar la base seleccionada
    escribe
       a la raíz del exponente seleccionado es: ", memoria[[1]]];]
```

Logaritmo del anti-logaritmo de un número con base de otro número:

Los registros 6, 7 y 8 no se usan.

```
In[ ]:= f2[base_Integer, antilogaritmo_Integer] := Module[{}, cop[base, 2];
                                                    módulo
    cop[antilogaritmo, 3];
```

```
asi[0, 9];
ig[2, 9, "end0", "next"];
Label["next"]; ig[3, 9, "end0", "next1"];
Label["next1"]; asi[1, 10]; ig[3, 10, "end1", "next2"];
Label["end1"];
asi[0, 1] ;; Print["El logaritmo de 1 para cualquier base es: ", memoria[[1]]];
Goto["end"];
Label["next2"]; ig[2, 10, "end1.1", "next4"];
Label["end1.1"];
Print["El logaritmo de base 1 para un número distinto de 1 no está definido"];
Goto["end"];
Label["next4"]; mei[2, 3, "next3", "endM"];
Label["endM"];
asi[0, 1];
Print["Como la base es mayor que el antilogaritmo el resultado es: ", memoria[[1]]];
Goto["end"];
Label["next3"]; asi[1, 4]; cer[5];
Label["iter"]; suc[5]; mul[2, 4, 4]; ig[4, 3, "End", "Next"];
Label["Next"]; mei[4, 3, "iter", "End2"];
Label["End"];
cop[5, 1];
Print["El resultado del logaritmo del número seleccionado es: ", memoria[[1]]];
Goto["end"];
Label["End2"];
cop[5, 1];
pre[1, "aquíNoSaltaNunca"];
Print["El resultado del logaritmo del número seleccionado es: ", memoria[[1]]];
Goto["end"];
Label["aquíNoSaltaNunca"];
Label["end0"];
```

```
    asi[0, 1];
    Print["El logaritmo de base 0 o de antilogaritmo 0 no está definido"];
    escribe
    Goto["end"];
    ve a
    Label["end"];
    etiqueta
   ]
```

Raíz cuadrada de un número:

Los registros 7, 8, 9 y 10 no se usan.

```
In[ ]:= f3[pos_Integer] := Module[{}, cop[pos, 2];
                          módulo
    cer[3];
    cer[4];
    cer[5];
    cer[6];
    ig[2, 6, "end", "iter"];
  Label["iter"];
  etiqueta
    cop[4, 6];
    suc[3];
    suc[4];
    mul[3, 4, 5];
    mei[5, 2, "iter", "end"];
  Label["end"];
  etiqueta
    cop[6, 1];
    Print["La raíz cuadrada entera inferior del número seleccionado es: ", memoria[[1]];]
    escribe
```

Función que eleva un número a la potencia de otro número dependiendo de su tamaño:

Los registros 5, 6, 7, 8 y 9 no se usan.

```
In[ ]:= f4[numeroN_Integer, numeroM_Integer] :=
         Module[{}, mei[numeroN, numeroM, "NelevadoaM", "MelevadoaN"];
          módulo
        Label["NelevadoaM"];
        etiqueta
          cop[numeroN, 2];
          cop[numeroM, 3];
          cer[10];
          ig[3, 10, "next", "next2"];
        Label["next"]; asi[1, 1]; Goto["end"];
        etiqueta                  ve a
        Label["next2"]; ig[2, 10, "next3", "next4"];
        etiqueta
        Label["next3"]; asi[0, 1]; Goto["end"];
        etiqueta                   ve a
        Label["next4"]; asi[1, 4];
        etiqueta
        Label["iter"]; pre[3, "end"]; mul[2, 4, 4]; cop[4, 1]; Goto["iter"];
        etiqueta                                               ve a
        Label["MelevadoaN"];
        etiqueta
          cop[numeroM, 2];
          cop[numeroN, 3];
          cer[10];
          ig[3, 10, "Next", "Next2"];
                    siguiente
        Label["Next"]; asi[1, 1]; Goto["end"];
        etiqueta  siguiente       ve a
        Label["Next2"]; ig[2, 10, "Next3", "Next4"];
        etiqueta
        Label["Next3"]; asi[0, 1]; Goto["end"];
        etiqueta                   ve a
        Label["Next4"]; asi[1, 4];
        etiqueta
        Label["Iter"]; pre[3, "end"]; mul[2, 4, 4]; cop[4, 1]; Goto["Iter"];
        etiqueta                                               ve a
        Label["end"];
        etiqueta
          Print["El resultado de elevar la base seleccionada
          escribe
              a la raíz del exponente seleccionado es: ", memoria[[1]]];]
```