

CMC-Tema-4-Maquinas-de-registros...



Anónimo



Computabilidad y complejidad



3º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingeniería Informática Universidad Politécnica de Valencia



Que no te escriban poemas de amor cuando terminen la carrera



WUOLAH





No si antes decirte Lo mucho que te voy a recordar

(a nosotros por suerte nos pasa)

Apuntes CMC – Tema 4: Máquinas de Registros

Generalidades

En este tema describiremos diferentes modelos de cálculo basados estructuralmente en los computadores digitales ordinarios.

Estos modelos consisten en máquinas de registros que abstraen e idealizan los elementos constituyentes de los computadores digitales, de modo que, por su idealización, no tienen existencia real y permiten, al igual que la máquina de Turing, definir la familia de las funciones computables. De hecho todos los modelos que describiremos son computacionalmente equivalentes al modelo de Turing.

Los modelos descritos a continuación son:

- El modelo RAM
- El modelo RASP
- El modelo de la máquina contador

El modelo RAM

La RAM (Random Access Machine) se compone de :

- Una memoria consistente en un número ilimitado de registros. Cada uno de ellos tiene asociada una dirección consistente en un número natural mediante el cual se puede directamente acceder a su contenido: un número natural. El conjunto de direcciones coincide con el conjunto de los números naturales.
- Una unidad de control donde se encuentra el programa a ejecutar junto con el contador de programa. El programa se compone de una secuencia finita de instrucciones pertenecientes a un conjunto finito de instrucciones muy básicas y sencillas.
 - Este modelo de máquina tiene instrucciones con direccionamiento indirecto. Las instrucciones se encuentran numeradas consecutivamente empezando en 0. La ejecución siempre comienza en la primera instrucción y si termina lo hace cuando se intenta acceder a una instrucción inexistente.
- La arquitectura de una RAM se corresponde con la arquitectura de Harvard.
- (Nota) El programa está fuera de la memoria.
- Cada registro será designado mediante la notación: Ri, i ∈ N; donde i es su dirección. Su contenido lo denotaremos mediante: [i].
- Esta máquina la utilizaremos para computar funciones numéricas de la forma:

$$g: N^m \rightarrow N^k$$

- Para su cómputo la máquina arranca con los *m* primeros registros con los argumentos de la función y el resto de registros con valor nulo. Cuando la máquina termine el cómputo, su resultado será el contenido de los *k* primeros registros; en otro caso, si el cómputo no termina el resultado no está definido.
- Así, la máquina computa la función g definida de modo que...:

$$g(n_1, ..., n_m) = (j_1, ..., j_k)$$

Siempre que la máquina arranca con:

v termina con:

■ [i] = n_i + 1, i = 0, ..., m-1,

[i] =
$$j_i$$
+1, i = 0, ...,k-1

• [i] = 0, ∀ i ≥ m



El modelo RASP

La RASP (Random Access Stored Program Machine) es como una RAM pero con el programa a ejecutar almacenado en la propia memoria; así su arquitectura se corresponde con la de Von Neumann.

Es una máquina universal que puede simular a cualquier RAM y en consecuencia realizar todas sus computaciones. Es en este sentido similar a la máquina de Turing Universal (véase el tema 5).

La Máquina Contador

El modelo de la *Máquina Contador* es una restricción de la *RAM*. Sólo dispone de un número finito de registros (que naturalmente pueden variar de una máquina a otra) y no tiene instrucciones con direccionamiento indirecto.

- → El resto de esta sección lo dedicaremos a su estudio exponiendo diversos ejemplos operativos
- Este modelo de máquina, aunque presente restricciones, tiene el poder computacional de la RAM y de la RASP.

Su conjunto básico de instrucciones es el siguiente:

• suc(i), su semántica es la siguiente: incrementa en una unidad el contenido de Ri, es decir:

La ejecución continúa en la siguiente instrucción del programa, si no, termina.

- **pre(i, k)**, su semántica es la siguiente:
 - Si [i] > 0, entonces [i] ← [i] 1; la ejecución continúa en la siguiente instrucción del programa si existe, si no termina;
 - En otro caso, la ejecución continúa en la **k-ésima** instrucción si existe, si no termina.

Este par de instrucciones, junto con la existencia de un registro predefinido con contenido nulo son suficientes para realizar cualquier computación.

Seguidamente, veremos algunos fragmentos de programa que ejecutan cálculos sencillos que ilustran la operatividad de este modelo.

En lo inmediato que sigue **Rc** tiene asignado permanentemente el valor **nulo**.

```
[i] ← 0
    n : pre (i, n + 2)
    n + 1: pre (c, n)

[i] ← [i] + [j] AND [j] ← 0 AND Ri!= Rj
    n : pre (j, n + 3)
    n + 1: suc (i)
    n + 2: pre (c, n)
```





(a nosotros por suerte nos pasa)

Ayer a las 20:20

Oh Wuolah wuolitah Tu que eres tan bonita

Siempres me has ayudado Cuando por exámenes me he agobiado

Llegó mi momento de despedirte Tras años en los que has estado mi lado.

Pero me voy a graduar. Mañana mi diploma y título he de pagar

No si antes decirte Lo mucho que te voy a recordar

Envía un mensaje...



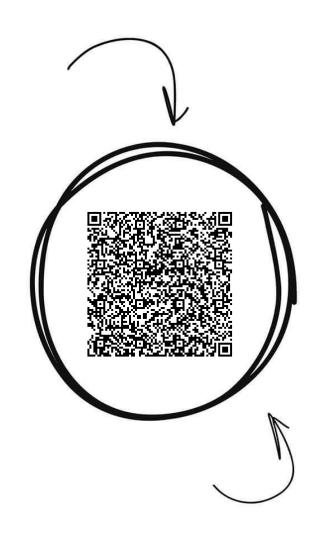








Computabilidad y complejidad



Note bank of the UPV





Comparte estos flyers en tu clase y consigue más dinero y recompensas

- Imprime esta hoja
- 2 Recorta por la mitad
- Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
- Llévate dinero por cada descarga de los documentos descargados a través de tu QR



Apuntes CMC - Tema 4: Máquinas de registros

Para evitar el tener que mantener un registro predefinido permanentemente con valor nulo, añadiremos al conjunto de instrucciones la instrucción:

- **goto (n)** : su semántica es la siguiente: ejecutar incondicionalmente la instrucción n-ésima si ésta existe, si no terminar la ejecución.
- → Las instrucciones **suc(i)**, **pre(i, k)** y **goto(n)** son suficientes para realizar cualquier computación.

Los ejemplos anteriores, utilizando la sentencia goto(n), quedan como sigue:

```
• [i] ← 0

n : pre (i, n + 2)

n + 1: goto (n)
```

n + 2 : goto(n)

Este Código puede simbólicamente referenciarse mediante la macroinstrucción cer (i)

```
• [i] ← [i] + [j] AND [j] ← 0 AND Ri!= Rj

n : pre (j, n + 3)

n + 1: suc (i)
```

• [i] ← k

cer (i)

n + 1: suc (i)

...

n + k: suc (i)

Este Código puede simbólicamente referenciarse mediante la macroinstrucción asi (k,i) (asignar al registro i el valor k).

• [i] \(\(\) AND Ri != Rj

Sea Rk un resigtro auxiliar de cáculo distinto de los anteriores: Ri != Rk AND Rj != Rk

```
cer (i)
cer (j)
n: pre(j,n+4)
n+1: suc(i)
n+2: suc(k)
n+3: goto(n)
n+4: pre(k,n+7)
n+5: suc(j)
n+6: goto(n+4)
```

Este Código puede simbólicamente referenciarse mediante la macroinstrucción cop (j,i) (copia el contenido del registro j en el registro i).







No si antes decirte Lo mucho que te voy a recordar

Pero me voy a graduar. Mañana mi diploma y título he de pagar

(a nosotros por suerte nos pasa)

Apuntes CMC – Tema 4: Máquinas de registros

• $[m] \leftarrow [p] + [q]$

Sean Ri y Rj dos registros auxiliares de cálculo distintos de los anteriores.

Este Código puede simbólicamente referenciarse mediante la macroinstrucción sum (p,q,m)

• [r] - [p] * [q]

Sean Ri, Rj, Rk y Rm cuatro registros auxiliares de cálculo distintos de los anteriores (se puede prescindir de Rj).

	cer(m)
	cer(k)
	cop(p,i)
	cop(q,j)
n:	pre(i,n+8)
n+1:	pre(j,n+5)
n+2:	suc(m)
n+3:	suc(k)
n+4:	goto(n+1)
n+5:	pre(k,n)
n+6:	suc(j)
n+7:	goto(n+5)
n+8:	cop(m,r)

Este Código puede simbólicamente referenciarse mediante la macroinstrucción mul (p,q,m)

Equivalentemente, utilizando la macroinstrucción para la suma:



• [r] ← [p] / [q] (División entera)

Sean Ri, Rj, Rk y Rm cuatro registros auxiliares de cálculo distintos de los anteriores.

```
n+5: qoto (n+2)
     cer(m)
                        n+6: suc(m)
     cer(k)
                        n+7: pre(k, n+2)
     cop(p,i)
                        n+8: suc(j)
     cop(q,j)
                        n+9: qoto(n+7)
  n: pre(j,error)
                       n+10: cop(m,r)
n+1: suc(j)
n+2: pre(j, n+6)
n+3: pre(i, n+10)
n+4: suc(k)
                        error:....
```

Este Código puede simbólicamente referenciarse mediante la macroinstrucción $\operatorname{div}(p,q,r)$

Sean Ri y Rj dos registros auxiliares de cálculo distintos de los anteriores.

```
cop(p,i)
cop(q,j)
n: pre(i,m1)
n+1: pre(j,m2)
n+2: goto(n)
```

Este Código puede simbólicamente referenciarse mediante la macroinstrucción mei <math>(p, q, m1, m2)

Sean Ri y Rj dos registros auxiliares de cálculo distintos de los anteriores.

```
cop(p,i)

cop(q,j)

n: pre(i,n+3)

n+1: pre(j,m2)

n+2: goto(n)

n+3: pre(j,m1)

n+4: goto(m2)
```

Este Código puede simbólicamente referenciarse mediante la macroinstrucción igu (p, q, m1, m2)



- while ([i] > 0) CÓDIGO (No es una macroinstrucción)
 - Si utilizamos while --> Tenemos que preocuparnos de que dentro de la parte de CÓDIGO modificamos el valor de i. n: pre (i, m+1)

Equivalencias entre modelos

Sea una función (parcial) de la forma:

$$g: N^m \longrightarrow N^k$$

Diremos que la función g es:

- RAM Computable si y sólo si existe una RAM que la compute.
- RASP Computable si y sólo si existe un programa para la máquina RASP con el que esta máquina puede computarla
- MC Computable si y sólo si existe una máquina contador que la compute.

<u>Teorema</u> Sea una función (parcial) de la forma:

$$q: N^m \longrightarrow N^k$$

Los siguientes enunciados son equivalentes:

- g es RAM computable
- g es RASP computable
- g es MC computable
- g es Turing computable

Sea una función (parcial) de la forma:

$$q: \mathbb{N} \longrightarrow \mathbb{N}$$

Diremos que la función g es: MC2 Computable si y solo si existe una máquina contador con dos contadores que la compute.

<u>Teorema</u> Sea una función (parcial) de la forma:

$$q: \mathbb{N} \longrightarrow \mathbb{N}$$

Los siguientes enunciados son equivalentes:

- *g* es *MC* computable
- g es MC2 computable

<u>Observación</u>: El poder computacional del modelo de la máquina contados con dos contadores también equivale al de los modelos:

- RAM - RASP - Turing

