- Se va a trabajar sobre una matriz

- La matriz la escalamos por un factor K
  -> Entonces también sus indices se escalan
- El algoritmo que se sigue es:

- Pixeles horizontales y verticales

  - La fórmula general para cada ramo desconocido es:

$$\text{valor} = \frac{\text{Siguiente indice conocido} - \text{indice actual}}{\text{Siguiente indice conocido} - \text{pasado indice conocido}} \times \text{Pasado valor conocido} + \frac{\text{indice actual} - \text{pasado indice conocido}}{\text{Siguiente indice conocido} - \text{pasado indice conocido}} \times \text{Siguiente valor conocido}$$

Esta vara va a dar punto flotante
L> Drop the floating

- Pixeles intermedios:
  - Se calcula exactamente igual que los pixeles horizontales, pero se debe hacer después porque depende del paso anterior.

- Limitaciones de la imagen:
  - Dimensión: 390x390
  - Escala de grises: [0,255] -> 2 bytes · 65535
  - El algoritmo se aplica sobre una sección de: 98 x 98 bits

ISA:

- Modos de direccionamiento:
  - Vamos a basarnos sobre una RISC, de donde los modos posibles son
    - Offset -> LD R1, R2 + Dx3
    - Index -> LD R1, R2+R3
    - Pc relative -> Branching
  - Almacenamiento interno: reg - reg

- Tamaño y tipo de datos:
  - Cada píxel puede representarse por un byte
  - Debido a las multiplicaciones podemos presentar overflow: 2 bytes máx
  - Los contadores pueden ser de modo 98 bits + 14 = 112 bits
    - Mínimo
    - Siguiente entero de 2 bytes
  - Tipo de datos:
    - pixel: 1 byte
    - bixel: 2 bytes

- Tipo y sintaxis de las instrucciones:
  - Lista de instrucciones:

Aritméticas lógicas:
1. Suma — Sum dest, op1, op2
2. Resta — res dest, op1, op2
3. Multiplicación — mul dest, op1, op2
4. División — div dest, op1, op2
5. AND — and dest, op1, op2
6. OR — o- dest, op1, op2
7. NOT — not dest, op1, op2

Control Flujo:
8. Salto — sto dest
9. Salto menor — sme dest
10. Salto igual — si dest
14. Salto mayor — sma dest

Aritmética / Transferencia de datos:
11. Comparación — cmp op1, op2
12. Carga de memoria — ld dest, dir
13. Store memoria — sr op, dir

- Codificación
  - 13 instrucciones:
    - 7 tres operandos - 3 un operando - 3 dos operandos

    xxxx } 4 bits para 14 instrucciones

  - Inmediato de máximo 1 byte: -> Son las operaciones las que pueden dar overflow

    xxxx xxxx

  - 16 registros: xxxx } - 4 bits

  - Tamaño de palabra:
    - Tiene que ser fijo
    - Tres operandos:

      xxxx xxxx xxxx xxxxxxxx : 24 bits
      23 19 15 11 7 0

      xxxx xxxx xxxx xxxxxxxx xxxx
      11 4

    - Dos operandos:

      xxxx xxxx xxxx xxxxxx xxxxx -> Reg, Reg

      xxxx xxxx xxxx xxxxxx xxxx -> Reg, Imm

      ↳ Indicador inmediato

      ↳ 1: El segundo operando es inm

      0: El segundo operando es reg

    - Load/Store con offset/index:

      xxxx xxxx xxxx xxxxxxxx xx1x

      ↳ Indica L/S

      ↳ 0: offset

      1: Index

    - Un operando:

      xxxx xxxx xxxxxxxxxxxxxxxx

- Tablas de verdad

• Instrucción

0 0 00 -> Suma
0 0 01 -> Resta
0 0 10 -> Multiplicación
0 0 11 -> División
0 1 00 -> AND
0 1 01 -> OR
0 1 10 -> NOT

Tres operandos

Tabla ALU

1 0 00 -> cmp
1 0 01 -> ld
1 0 10 -> sr

Dos operandos

1 1 00 -> sto
1 1 01 -> sme
1 1 10 -> si
1 1 11 -> sma

Un operando

- De lo anterior, podemos descomponer el campo de instrucción así:

a b c d

- Utilizando la siguiente lógica:
  - Tres operandos: $\bar{a}$
  - Dos operandos: $a\bar{b}$
  - Un operandos: ab

- Registros
  1. R0
  2. R1
  3. R2
  4. R3
  5. R4
  6. R5
  7. R6
  8. R7
  9. R8
  10. R9
  11. R10
  12. R11 -> Flags: zero, neg, ov, carry
  13. R12 -> Sb
  14. R13 -> Sp
  15. R14 -> LR
  16. R15 -> PC

- Diseño modular
  - Decode
    - Mux A:
      - Es instrucción de tres operandos? ($\overline{Instr[23]}$)

        Modo ALU = { Instrucción [22:20) , sí
        Mux E , no
    - Mux B:
      - Hay un inmediato? (Instr [0])

        Operando 2 de la ALU = { Inmediato , sí
        Datos del registro en Instr [11:8] , no

    - Mux C:
      - Es instrucción de tres operandos? ($\overline{Instr[23]}$)

        Dirección del registro para el operando 1 de la ALU = { Instr [11:8] , sí
        Instr [15:12] , no

    - Mux D:
      - Es instrucción de tres operandos? ($\overline{Instr[23]}$)

        Operando 1 de la ALU = { Datos del registro en Instr [19:16) , sí
        Datos del registro en MuxC , No

    - Mux E:
      - Es instrucción Load o Store? (Instr [1])

        Alternativa Mux E = { 000 (Suma) , sí
        001 (Resta) , No

Conflicto, nunca pasa $\overline{Mux\ C}$ ^ Mux D al ser misma condición