Arquitectura de Computadores
Problemas (hoja 3). Curso 2025-26

Dado un procesador con repertorio RV64GV, con las instrucciones que figuran a continuación, escribir un programa vectorial que se ejecute en tiempo mínimo para los siguientes códigos.

Nota: suponer que un float ocupa 32 bits. VLEN=128
Nota: se puede suponer que los registros escalares están inicializados (indicar a qué valor).

Instrucciones vectoriales RVV 1.0
Configuración:
vsetvli rd, rs1, vtypei
vsetivli rd, uimm, vtypei
Loads:
vle8.v vd, (rs1), vm # 8-bit unit-stride load. Igual para 16, 32 y 64
vlse8.v vd, (rs1), rs2, vm # 8-bit strided load
Stores:
vse8.v vs3, (rs1), vm # 8-bit unit-stride store. Igual para 16, 32 y 64
vsse8.v vs3, (rs1), rs2, vm # 8-bit strided store
Aritméticas, lógicas y desplazamientos:
vop.vv vd, vs2, vs1, vm # integer vector-vector vd[i] = vs2[i] op vs1[i] op=add, sub, rsub, and, or, xor, sll, srl, sra, mul, div
vop.vx vd, vs2, rs1, vm # integer vector-scalar vd[i] = vs2[i] op x[rs1]
vop.vi vd, vs2, imm, vm # integer vector-immediate vd[i] = vs2[i] op imm No existen mul ni div
vfop.vv vd, vs2, vs1, vm # FP vector-vector operation vd[i] = vs2[i] fop vs1[i] fop=fadd, fsub, frsub, fmul, fdiv
vfop.vf vd, vs2, rs1, vm # FP vector-scalar operation vd[i] = vs2[i] fop f[rs1]

Ejercicio 1 (Load y store con stride)

```
float A[16][64];
int j=1;

for (i=0; i < 16; i++) {
    A[i,j] = (A[i,j-1]+ 3* A[i][j]+A[i,j+1])/5;
  }
```

Ejercicio 2

```
void addVec( float A[N], float B[N], float C[N]) {
    int i;
    for (i=0;i<N;i++)
          C[i] = A[i]+B[i];
    return;
}
```

Ejercicio 3 (Generación de mascara y ejecución usando máscara)

```
void addVec2(int x[N], float A[N], float B[N], float C[N]) {
    int i;
    for (i=0;i<N;i++) {
       if (x[i]!=0)
          C[i] = A[i]+B[i];
    }
    return;
}
```

Ejercicio 4

```
int   min(int * vec, int  len) {
    if (len <= 0) return -1;
    int  min = vec[0];
    for (int i = 1; i < len; ++i) {
        if (vec[i] < min) {
            min = vec[i];
        }
    }
    return min;
}
```

Ejercicio 5

```
float vector_dot_product(float* lhs, float* rhs, int len) {
    float result = 0.0f;
    for (int i = 0; i < len; ++i) {
        result += lhs[i] * rhs[i];
    }
    return result;
}
```

Ejercicio 6

```
int A[32][256];
int B[32][32];
int C[32];

for (i=0; i < 32; i++) {
  {
    C[i] = (7*A[i][0]- 5*B[i][0])/3;
  }
```

Ejercicio 7

```
int A[4];
int B[4];
int C[4];
int resultado=0;
int i;

for (i=0; i < 4; i++) {
  {
resultado = resultado + B[i];
if (A[i] > 0) (A[i] = - A[i];
C[i] = A[i] + B[i];
  }
```