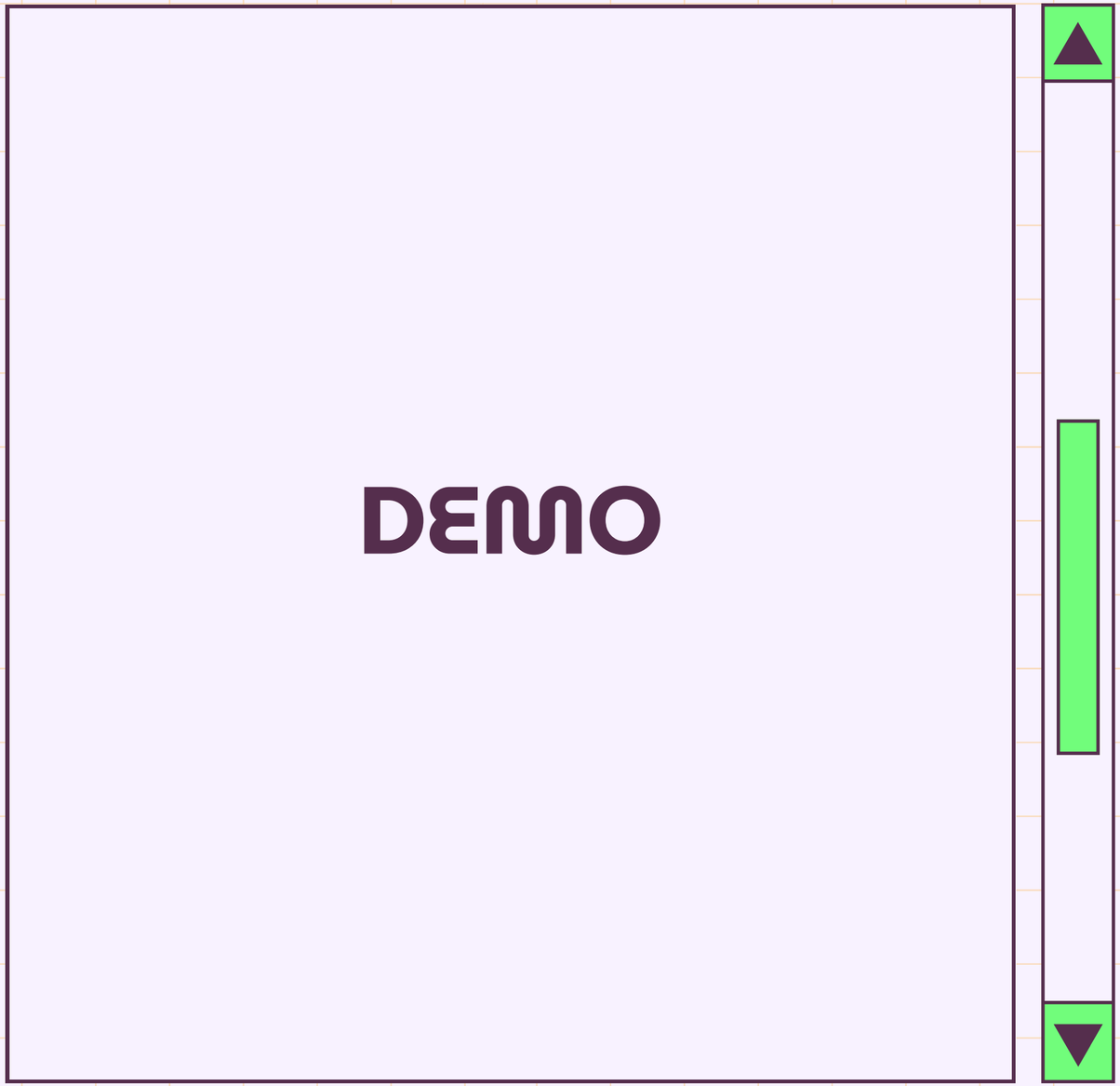
JS VS WASM

Grupo 11

01 Demo

02 Implementación

03 Resultados


DEMO

FLAGS

emcc factorizacion.c -o factorizacion.js -sEXPORTED_FUNCTIONS=_factorizacionPrima -sEXPORTED_RUNTIME_METHODS=UTF8ToString,ccall

IPLEMENTACIÓN WASM

factorizacion.c

factorizacion.js

factorizacion.wasm

```c
char* factorizacionPrima(const char* num) {
    // Parsea el char a un long long int
    long long int n = atoll(num);
    long long int divisor = 2;
    char* resultado = (char*)malloc(1000 * sizeof(char));

    while (n > 1) {
        if (n % divisor == 0) {
            if (esPrimo(divisor)) {
                char temp[100];
                sprintf(temp, "%lld ", divisor);
                strcat(resultado, temp);
            }
            n /= divisor;
        } else {
            divisor++;
        }
    }
  return resultado;
}
```

IPLEMENTACIÓN JS

js_factorizacion.js

```
function factorizacionPrima(n) {
  let divisor = 2;
  let result = "";

  while (n > 1) {
      if (n % divisor === 0) {
          if (esPrimo(divisor)) {
              result += `${divisor} `;
          }
          n /= divisor;
      } else {
          divisor++;
      }
  }
  return result;
}
```

IPLEMENTACIÓN HTML

index.html

```
function ejecutarJs() {
    let numeroJS = document.getElementById('numeroInput').value;
    if (numeroJS !== "") {
        const startJs = performance.now();
        const result = factorizacionPrima(numeroJS);
        const endJs = performance.now();
        const timeJs = (endJs - startJs) / 1000;
        document.getElementById('resultado-js').innerText = result + '\nTiempo de ejecución: ' + timeJs + ' segundos';
        resultadosJs.push(timeJs);
        myChart.update();
    }
    else {
        document.getElementById('resultado-js').innerText = 'Por favor ingrese un número';
    }
}
```

IPLEMENTACIÓN HTML

index.html

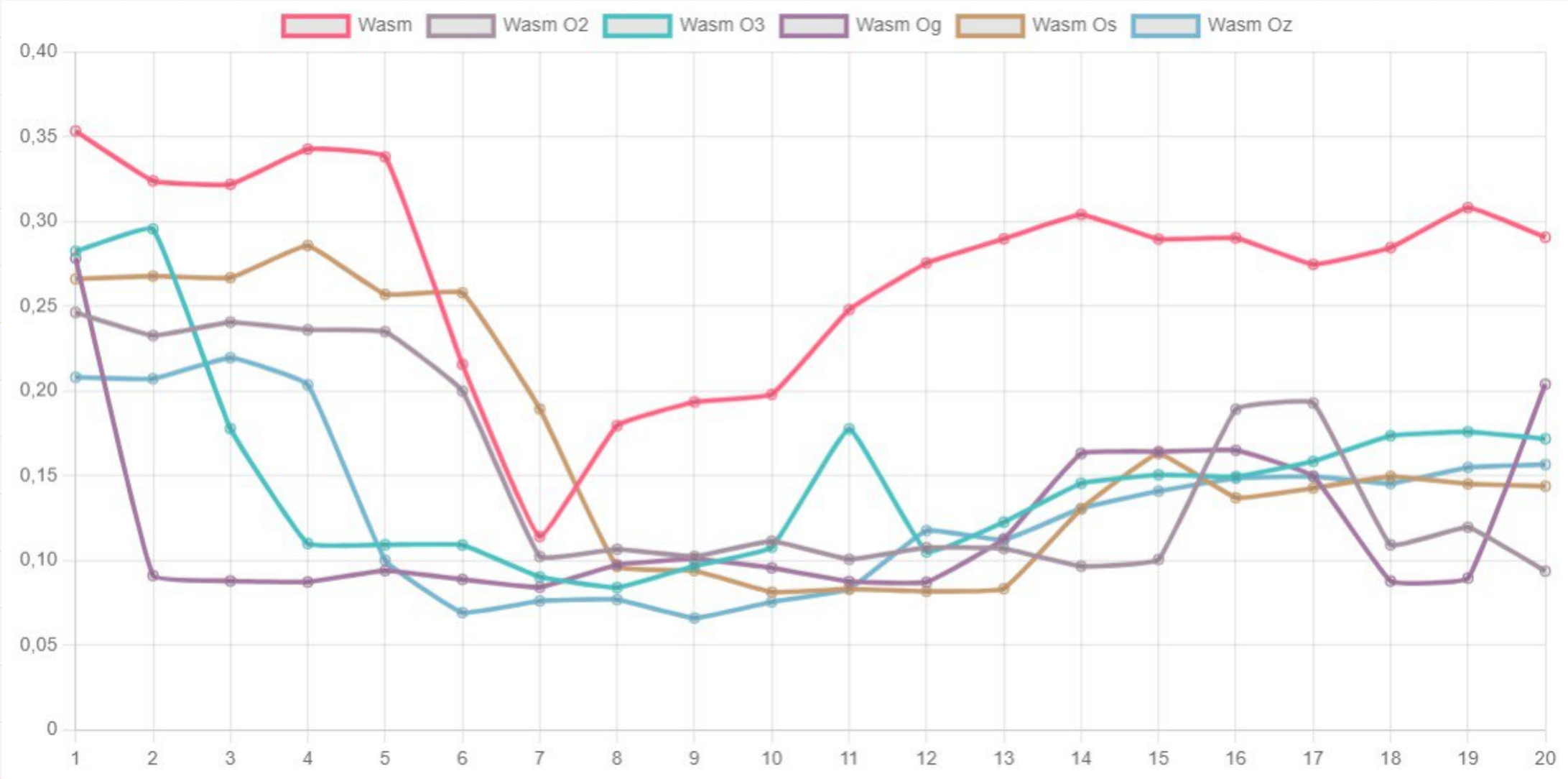
```
function ejecutarWasm() {
    let numeroWASM = document.getElementById('numeroInput').value;
    if (numeroWASM !== "") {
        const startWasm = performance.now();
        const resultado_ptr = Module.ccall('factorizacionPrima', 'number', ['string'], [numeroWASM.toString()]);
        const endWasm = performance.now();
        const result = Module.UTF8ToString(resultado_ptr);
        const timeWasm = (endWasm - startWasm) / 1000;
        document.getElementById('resultado-wasm').innerText = result + '\nTiempo de ejecución: ' + timeWasm + ' segundos';
        resultadosWasm.push(timeWasm);
        labels.push(resultadosWasm.length);
        myChart.update();
    }
    else {
        document.getElementById('resultado-wasm').innerText = 'Por favor ingrese un número';
    }
}
```

RESULTADOS

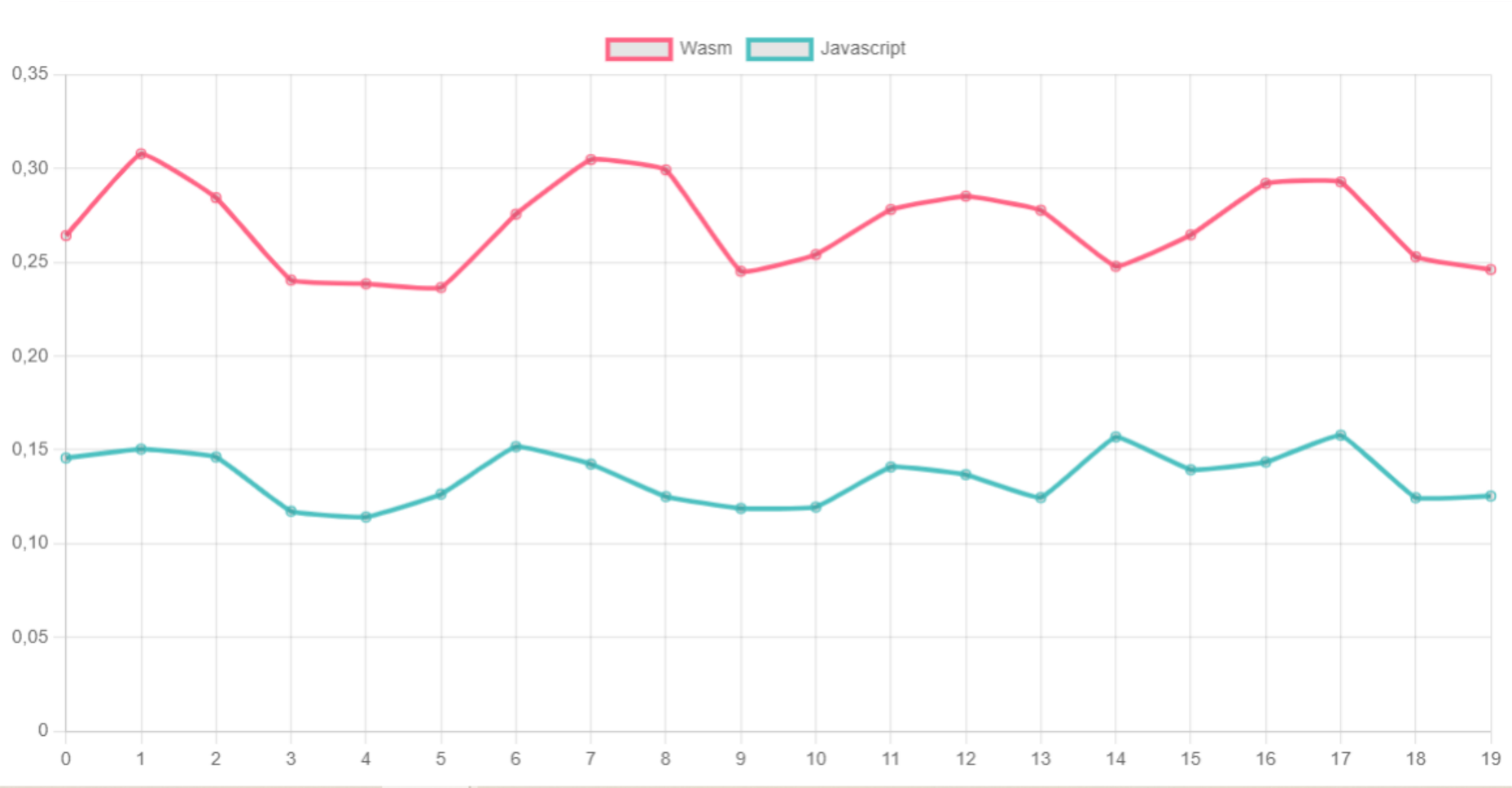
Tiempos de ejecución cuando los números primos resultantes son grandes:
30s vs 110s

23934879823
Números primos:
[19, 1259730517]

RESULTADOS OPTIMIZACIÓN



RESULTADOS



Tiempos de ejecución cuando los números primos resultantes son chicos

234567342
Números primos:
[2, 3, 3, 13031519]

OBSERVACIONES

01 Javascript es igual o más rápido que C en números resultantes pequeños

02 Javascript tiene tiempos de ejecución más variados: caché y optimización del navegador

03 Para problemas con más computo, C es mejor

04 No hay mucha diferencia entre las optimizaciones