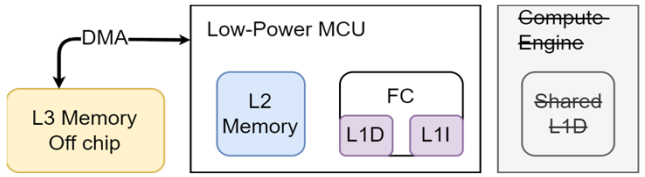
Efficient *Matrix Multiplications* on RISC-V MCUs

Paper: Cristian Ramírez, Adrián Castellón, Enrique S. Quintana-Ortí.
Poster: Cristian Ramírez, Jie Lei

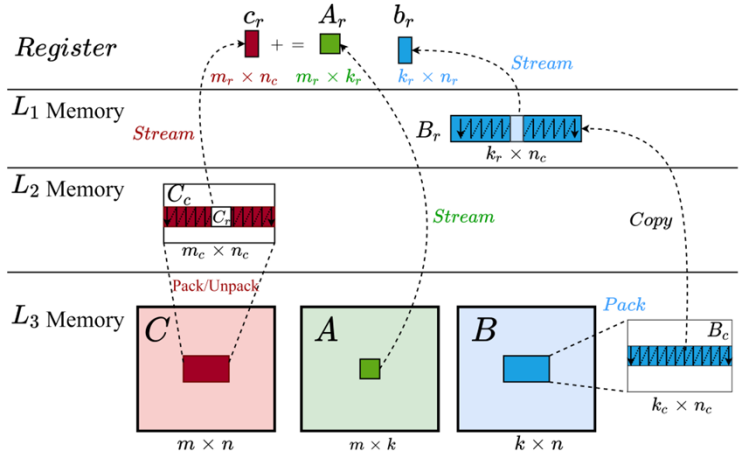
Universitat Politècnica de València

- GAP8
- Leverage Dot product **vector instructions**
- New Multi-level **memory hierarchy**
- Approximate Computing
- Modified BLIS (BLAS-like Library Instantiation Software)

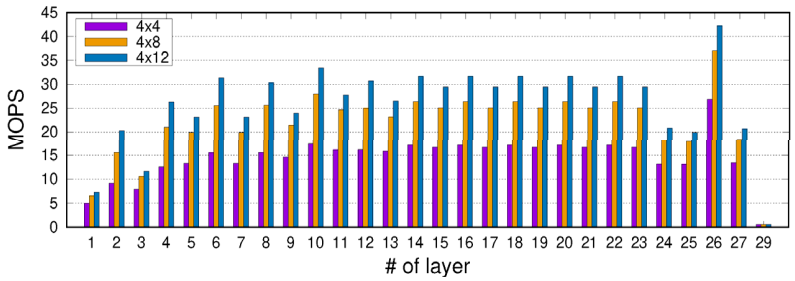
GAP8 **Memory** Architecture



Matrix Packing, Copying, Streaming



Performance on Different **Dimension of Microkernel** and **Convolutional Layers of MobileNet V1**



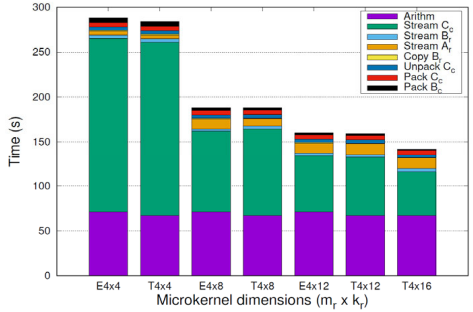
Proposed GEMM algorithm (B3C2A0)

```
L1   for ( jc = 0; jc < n; jc += nc )
L2     for ( pc = 0; pc < k; pc += kc ) {
         Bc ← B(pc : pc + kc − 1, jc : jc + nc − 1); // Pack
L3       for ( ic = 0; ic < m; ic += mc ) {
           Cc ← C(ic : ic + mc − 1, jc : jc + nc − 1); // Pack
L4         for ( pr = 0; pr < kc; pr += kr )
L5           for ( ir = 0; ir < mc; ir += mr )
L6             for ( jr = 0; jr < nc; jr += 1 ) // Micro-kernel
                 Cc(ir : ir + mr − 1, jr)
                   += Ac(ir : ir + mr − 1, pr : pr + kr − 1)
                    · Bc(pr : pr + kr − 1, jr);
           C(ic : ic + mc − 1, jc : jc + nc − 1) ← Cc; // Unpack
       }}
```

Proposed micro-Kernel (Loop 6)

```
gemm_ukernel_ABresident_gap8( int nc, signed char *A, int ldA,
                              signed char Br , signed char Cc )
{
 int jr, baseCB = 0;
 v4s A0, A1, A2, A3, // Columns of the 4x4 micro-tile Ar
     br, cr;         // Columns of Br, Cr

 // Load the columns of the 4x4 micro-tile Ar into vector
registers
 // Simulated in software using the v4s datatype
 A0 = ((v4s ) (&A[0]);       A1 = ((v4s ) (&A[ldA] );
 A2 = ((v4s ) (&A[2*ldA]);   A3 = ((v4s ) (&A[3*ldA] );

 // Transposition of Ar omitted for brevity
 // ...
 for ( jr = 0; jr < nc; jr++ ) { // Loop L6
   // Load the jr-th columns of Cr, Br into two vector registers
   cr = ((v4s ) (&Cr[baseCB]);
   br = ((v4s ) (&Br[baseCB]);

   // Update i-th entry of cr as cr[i] += Ai * br, i=0,1,2,3,
   // The GAP8 dot product is realized in software via the
   // gap8_dotp4 instruction
   cr[0] += gap8_dotp4(A0, br); cr[1] += gap8_dotp4(A1, br);
   cr[2] += gap8_dotp4(A2, br); cr[3] += gap8_dotp4(A3, br);

   // Store the column of Cr in memory. No vector support in
GAP8
   Cr[baseCB+0] = cr[0]; Cr[baseCB+1] = cr[1];
   Cr[baseCB+2] = cr[2]; Cr[baseCB+3] = cr[3];

           baseCB += 4; // Prepare for next iteration
 }
```

Runtimes on Different micro-**kernel dimensions**



This work was supported by the research project PID2020-113656RB-C22of MCIN/AEI/10.13039/501100011033. C. Ramírez is a "Santiago Grisolía" fellow supported by Generalitat Valenciana. Adrián Castelló is a FJC2019-039222-I fellow supported by MCIN/AEI/10.13039/501100011033. This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955558. The JU receives support from the European Union's Horizon 2020 research and innovation programme, and Spain, Germany, France, Italy, Poland, Switzerland, Norway.
Jie Lei is conducted within the project APROPOS. This project has received funding from the European Union's Horizon 2020 (H2020) Marie Sklodowska-Curie Innovative Training Networks H2020-MSCA-ITN-2020 call, under the Grant Agreement no 956090. This content displayed do not represent the opinion of the European Union, and the European Union is not responsible for any use that might be made of its content.