An Embedded Systems Programming Environment for C

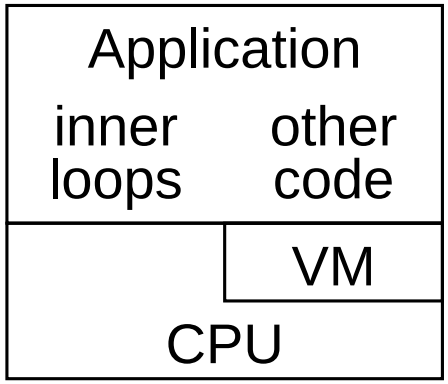
Bernd Burgstaller, The University of Sydney
Bernhard Scholz, The University of Sydney
M. Anton Ertl, TU Wien

Programming Embedded Systems

- Minimal resources: CPU speed, memory, power
- Wireless sensor networks: transfer size
- Programmed in assembly language or C
- Virtual Machine interpreter: small, but slow code
- Native machine code: fast, but large memory footprint

Mixed mode execution

combines the advantages



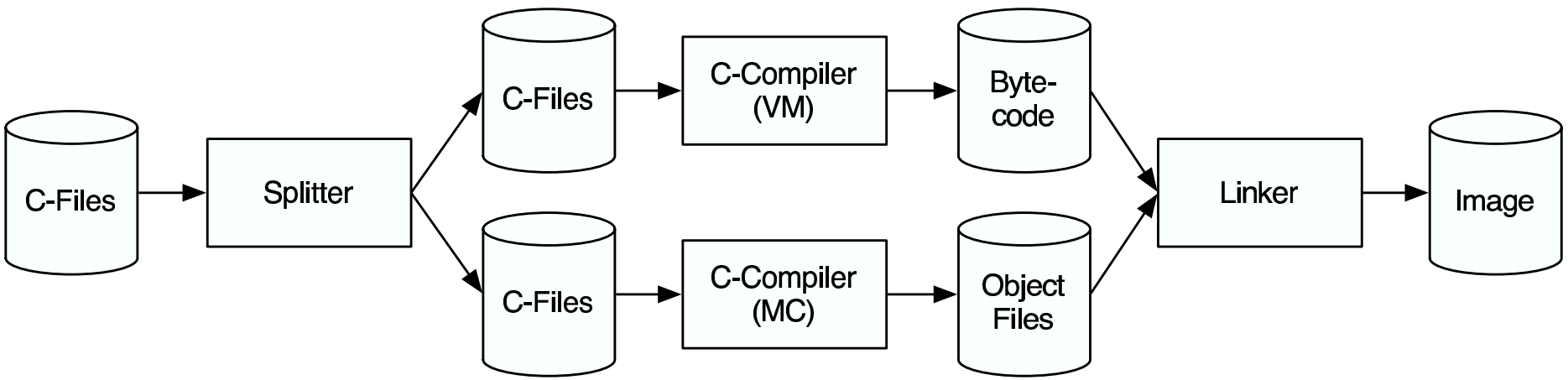
Source code division

Additional storage classes `vm` and `mc`

```
vm int main(int argc, char * argv[])
{
   ...
   fft(R_In, I_in, R_out, I_out);
   ...
}

mc void fft(float *R_In, float *I_in, float *R_out, float *I_out)
{
   ...
}
```

Compilation



Virtual Machine

- based on LCC intermediate representation
- stack-based, like Java VM
  - VM stack: local variables, computation
  - argument stack
  - Prog stack: for machine code
- implemented using Vmgen interpreter generator

Vmgen

- VM instruction specification:

```
add_i4 ( l1 l2 -- l )
l = l1+l2;
```

- generate VM interpreter, VM code generation support, VM disassembler, tracer, ...

- Performance: threaded code, top-of-stack caching, VM superinstructions

- Code compression: VM superinstructions

- Future work: Small encoding

Calls

VM-to-VM

```
addrl_p4 12288
arg_p4
addrl_p4 8192
arg_p4
addrl_p4 4096
arg_p4
addrl_p4 0
arg_p4
cnst_i4  4
call_v fft
```

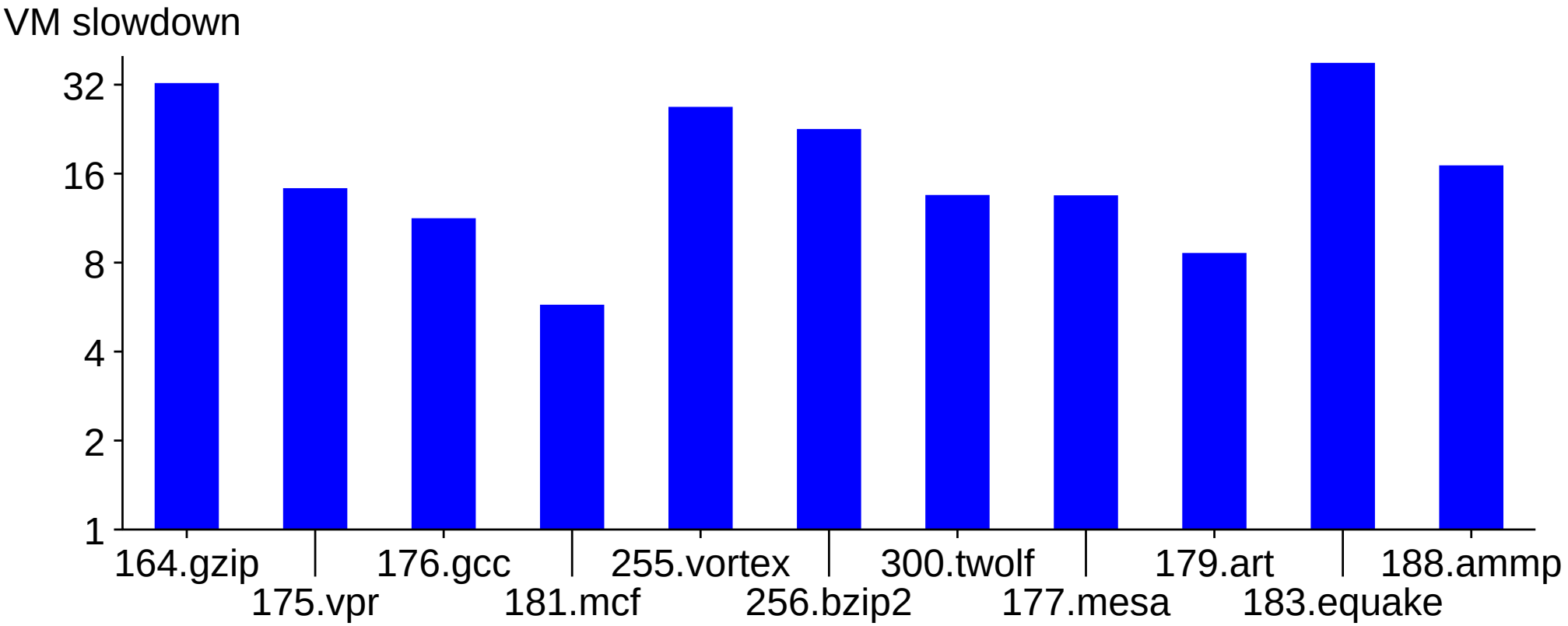
VM-to-MC

```
baddrg_p4 0 #table index
bcall_v
```

- MC-to-MC: ordinary call
- MC-to-VM: trampoline

Preliminary timing results

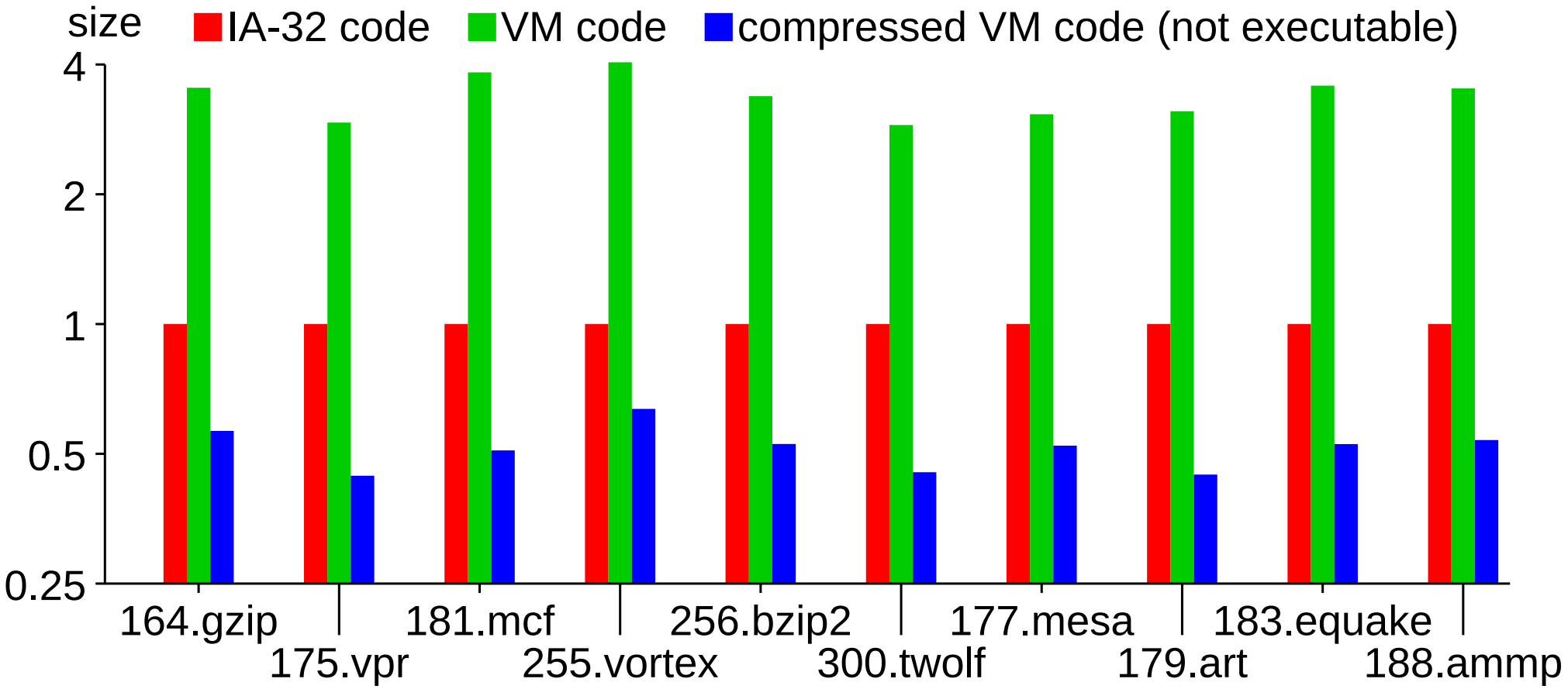
compared to machine code, on 1.8GHz Pentium 4, compiler: LCC
no VM superinstructions, no top-of-stack caching



preliminary superinstruction results: up to 3 times faster

Preliminary size results

no VM superinstructions, compiler: LCC



VM size on IA-32: 12KB

Conclusion

- Seamless integration of VM code and machine code
- Programmer decides and assigns storage classes `mc`, `vm`
- Switch between execution models on function boundaries
- Tools: LCC, Vmgen, Splitter, Linker
- VM alone: 5-38 times slower, currently larger memory footprint
- Transfer size smaller (factor of 2)
- Future work: Smaller encoding for VM, results for mixed mode