SV128: Scalable Addressing for PGAS Programming Models

John Leidel: Tactical Computing Labs

Steve Wallach: Micron

Overview

- Background
- Addressing Architecture
- Ongoing Research

background

HPC Jeopardy

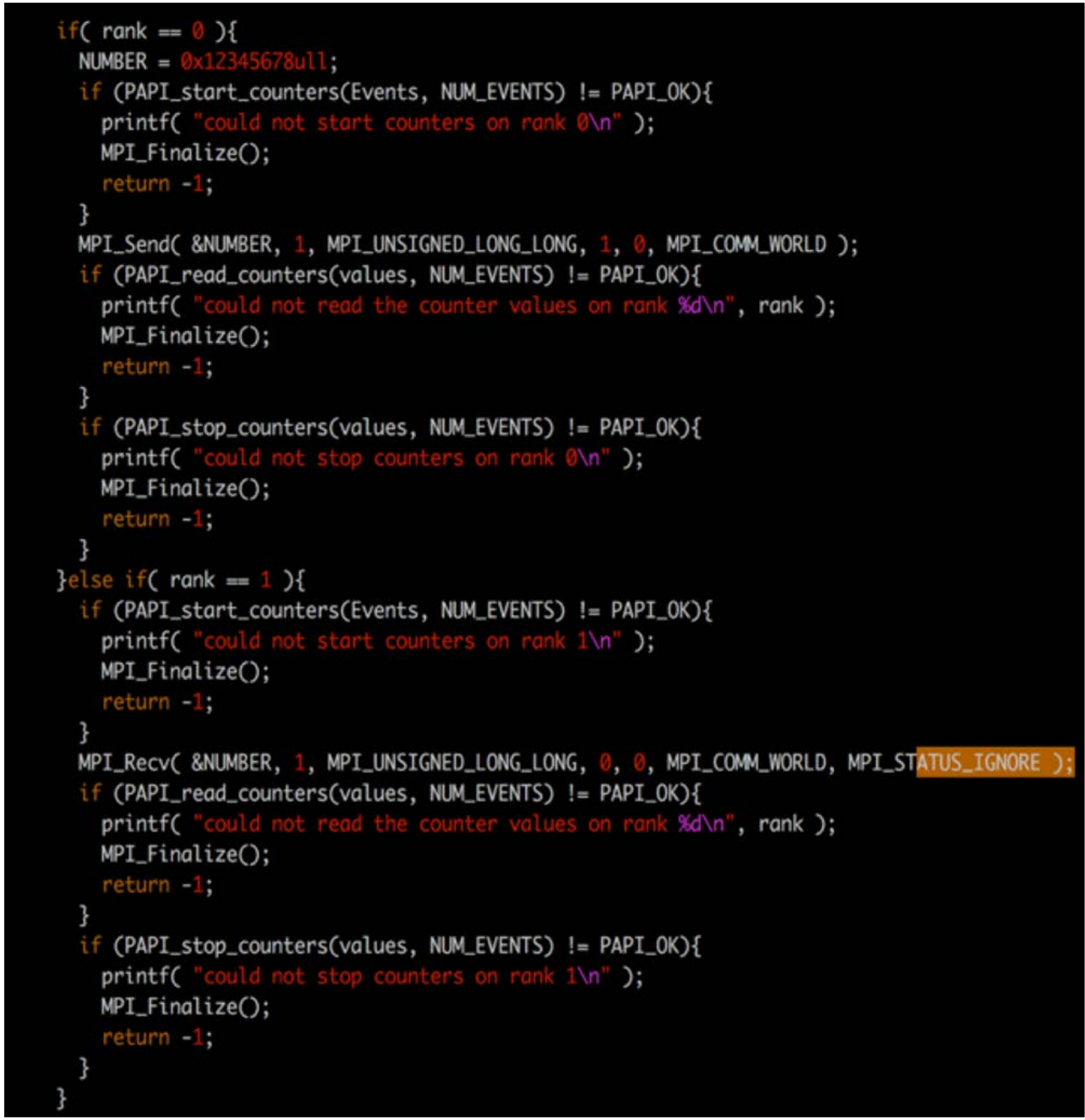
Ex-Cons	Instructions That Don’t Matter	Excuses for Linpack	PGAS Is Not a Medical Condition	Why didn’t I listen to my advisor?	Steve-isms
$100	$100	$100	$100	$100	$100
$200	$200	$200	$200	$200	$200
$300	$300	$300	$300	$300	$300
$400	$400	$400	$400	$400	$400
$500	$500	$500	$500	$500	$500

Lets have "PGAS is not a Medical Condition" for $500 Alex…

How expensive is a single transfer operation?

How expensive is it?

- Simple MPI Send/Recv
    - PAPI instruction counter
    - OpenMPI 1.10.2 (IB Verbs)
    - PAPI Git TOT (11/5/2018)
    - Ubuntu 16.04 4.4.0-87-generic
    - Intel E5620
- **1564 instructions!!**
    - **..each for Send/Recv**



```
if( rank == 0 ){
  NUMBER = 0x12345678ull;
  if (PAPI_start_counters(Events, NUM_EVENTS) != PAPI_OK){
    printf( "could not start counters on rank 0\n" );
    MPI_Finalize();
    return -1;
  }
  MPI_Send( &NUMBER, 1, MPI_UNSIGNED_LONG_LONG, 1, 0, MPI_COMM_WORLD );
  if (PAPI_read_counters(values, NUM_EVENTS) != PAPI_OK){
    printf( "could not read the counter values on rank %d\n", rank );
    MPI_Finalize();
    return -1;
  }
  if (PAPI_stop_counters(values, NUM_EVENTS) != PAPI_OK){
    printf( "could not stop counters on rank 0\n" );
    MPI_Finalize();
    return -1;
  }
}else if( rank == 1 ){
  if (PAPI_start_counters(Events, NUM_EVENTS) != PAPI_OK){
    printf( "could not start counters on rank 1\n" );
    MPI_Finalize();
    return -1;
  }
  MPI_Recv( &NUMBER, 1, MPI_UNSIGNED_LONG_LONG, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE );
  if (PAPI_read_counters(values, NUM_EVENTS) != PAPI_OK){
    printf( "could not read the counter values on rank %d\n", rank );
    MPI_Finalize();
    return -1;
  }
  if (PAPI_stop_counters(values, NUM_EVENTS) != PAPI_OK){
    printf( "could not stop counters on rank 1\n" );
    MPI_Finalize();
    return -1;
  }
}
```

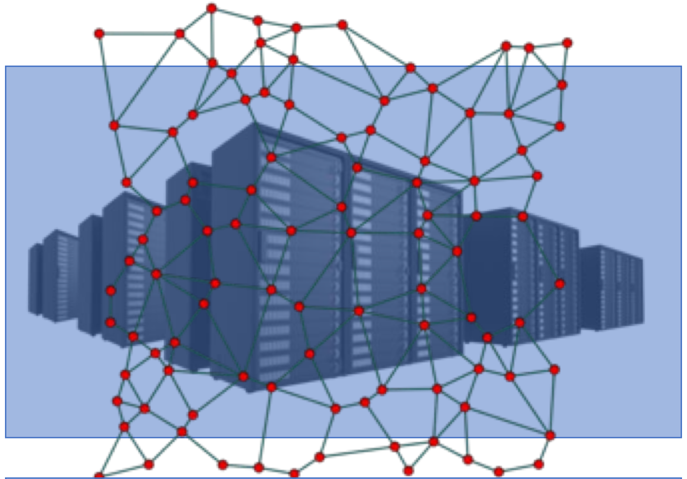
Problem: Need to define mechanisms for uArch global connectivity

- Distributed memory systems are here to stay
  - HPC
  - Cloud
  - Sensor nets
- How do we develop “enterprise” grade hardware infrastructure that promotes efficient, global execution?

- **Maximize the efficiency of application execution!**
- **Minimize the attack footprint for security-oriented operations!**

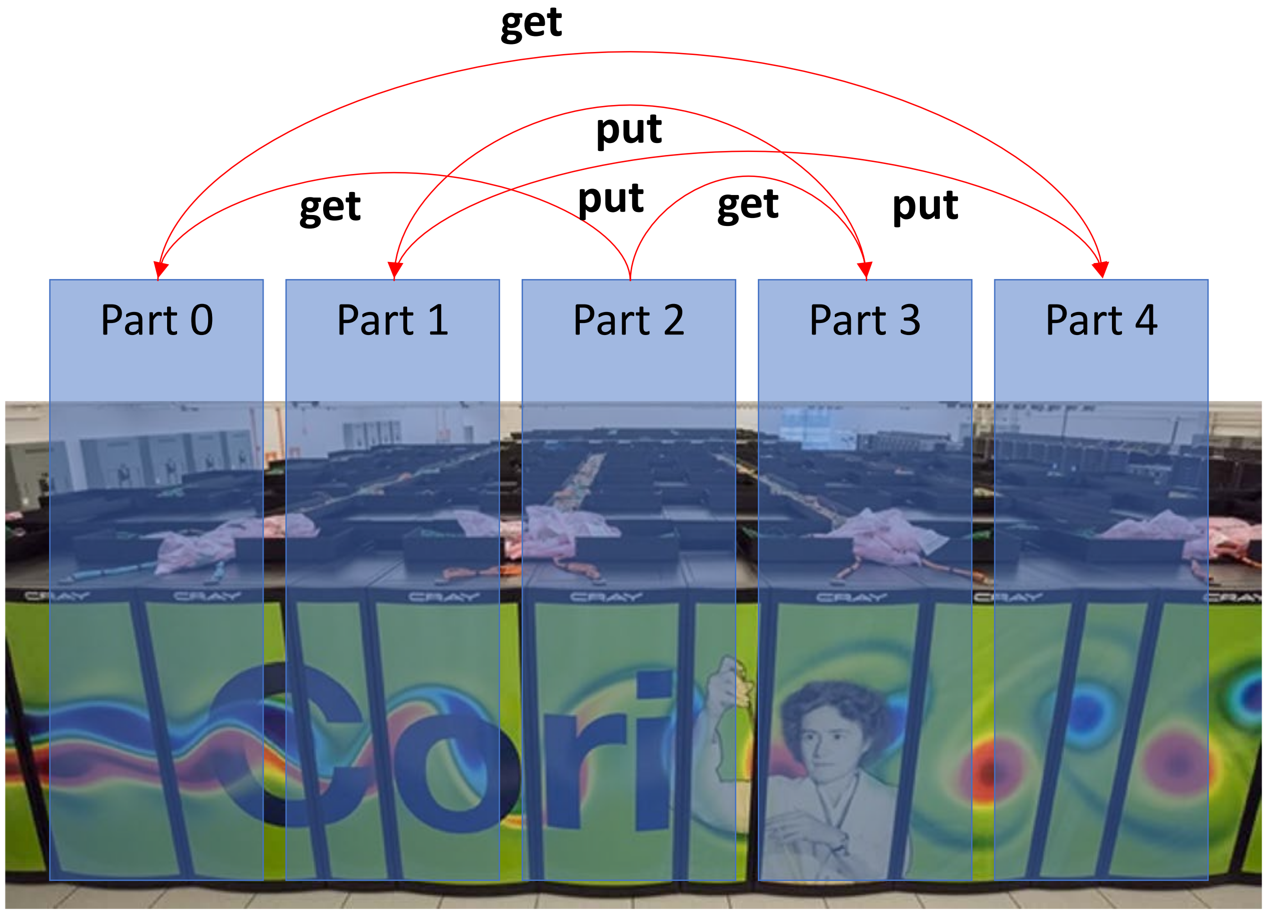
Application Domains

- HPA-FLAT
  - High performance analytics flat addressing
  - For extremely large datasets that are too difficult/time consuming to shard
- MMAP-IO
  - Map storage tiers into address space
  - Potential for object-based addressing
  - See DDN WOS
- Cloud-BSP
  - Potential for global object visibility for in-memory cloud infrastructures (Spark)
  - Reduce the time/cost to port Java to a full 128-bit addressing model
- Security
  - Fine grained, tagged security extensions to base addressing model
  - Tags are stored/maintained as ACL's for secure memory regions
- HPC-PGAS
  - High Performance Computing: Partitioned Global Address Space



HPC-PGAS

- Traditional message passing paradigm has tremendous amount of overhead
  - User library overhead, driver overhead
  - Optimized for large data transfers
  - Management of communication for Exascale-class systems
- We have excellent examples of low-latency PGAS runtimes, but little hardware/uArch support
  - LBNL: GASnet
  - PNNL: Global Arrays/ARMCI
  - Cray: Chapel
  - OpenSHMEM



xBGAS to SV128

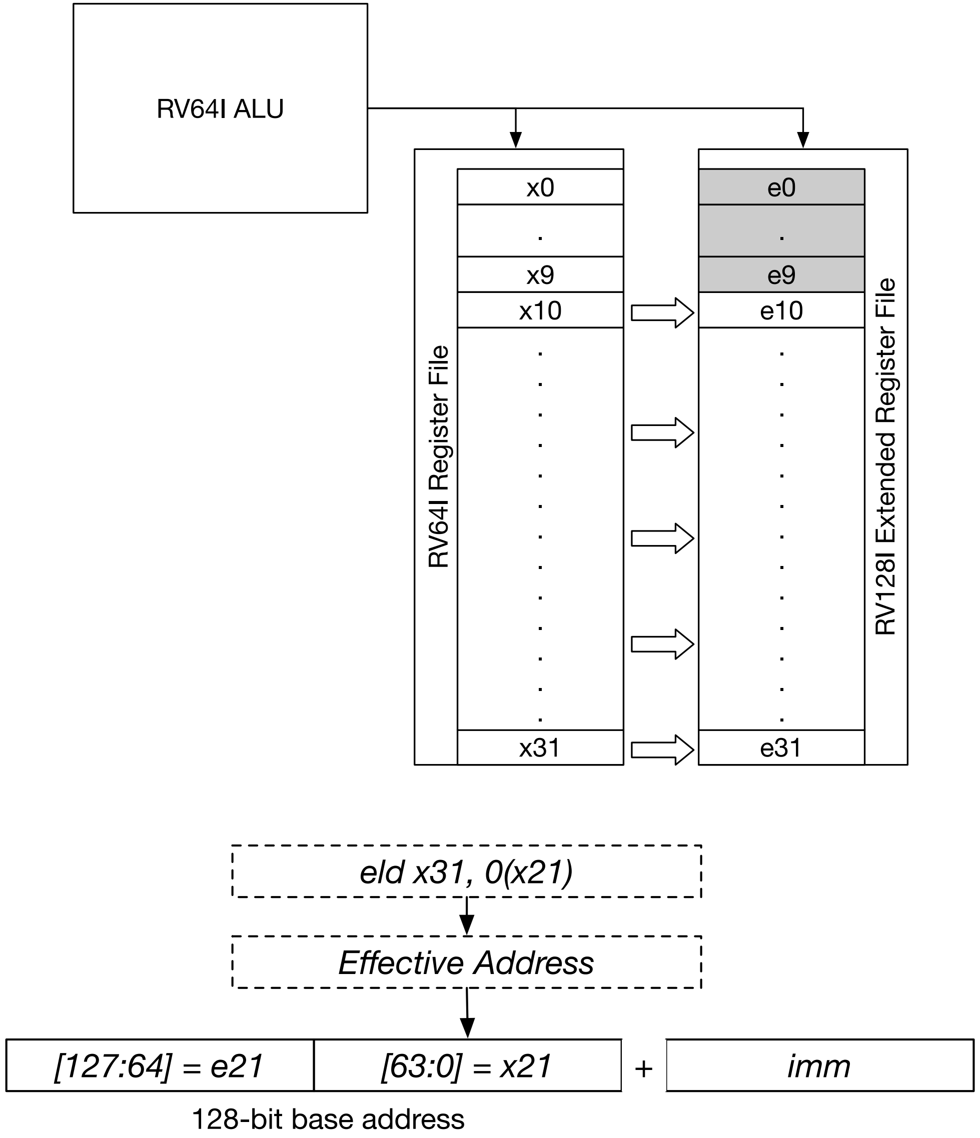
- Original xBGAS instruction set extension to RISC-V RV64 ISA
- In development for ~1 Year

- Now extended to include memory model: SV128
- RISC-V ISA spec that includes:
  - Security—enhanced addressing mechanisms
  - Extended memory addressing mechanisms (>64 bits of address space)
  - ISA extensions that make all this possible (looks & smells like RV64)
- Why RISC-V?
  - Its cheap and easy
  - SV128 NOT limited to RISC-V

addressing architecture

Addressing Architecture

- uArch maps extended addressing into RV64
    - We hope to generalize this for RV32 as well
- CSR bits encoded to appear as standard RV64 uArch
    - XLEN maps to RV64
    - TBD whether we need additional interrupts and exceptions
- Addition of *extended* {eN} registers that map to base general registers
- Extended registers are manually utilized via extended load/store/move instructions

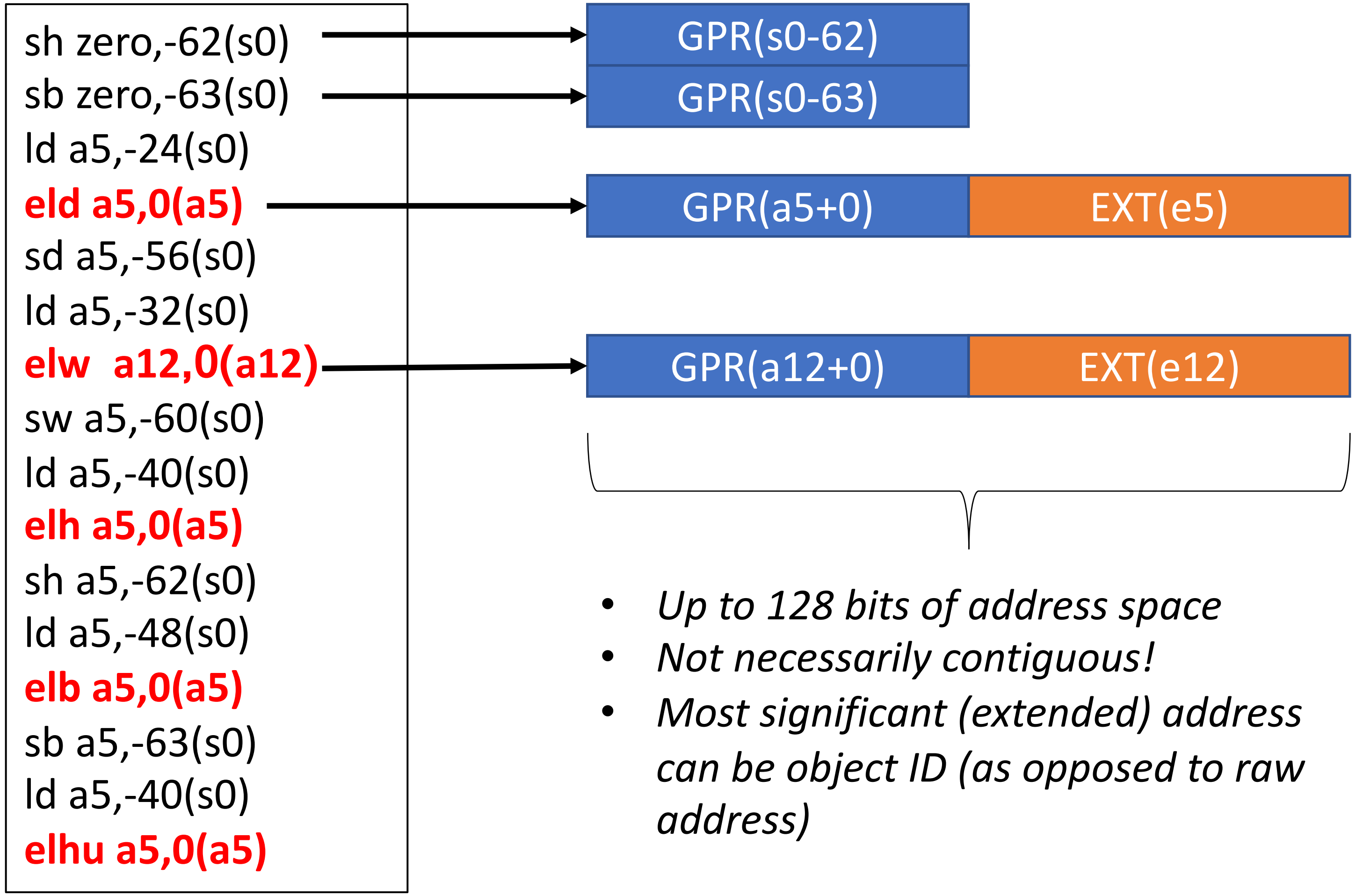


ISA Extension

- Instructions are split into three blocks:
  - Base integer load/store
  - Raw integer load/store
  - Address management
- Base integer load/store (I-type)
  - Permits loading/storing all base RV64I data types using standard mnemonic
  - EX: ***eld rd, imm(rs1)***
  - The extended register mapped to the same index as ’rs1’ is implied
- Raw integer load/store (R-type)
  - Permits loading/storing using explicit extended registers combined with explicit base registers (no imm)
  - ***erld rd, rs1, ext2***
  - LOAD(ext2[127-64], rs1[63-0])
- Address Management
  - Permits explicit manipulation of the extended register contents
  - ***eaddie extd, rs1, imm***
  - extd = rs1+imm

Addressing Example

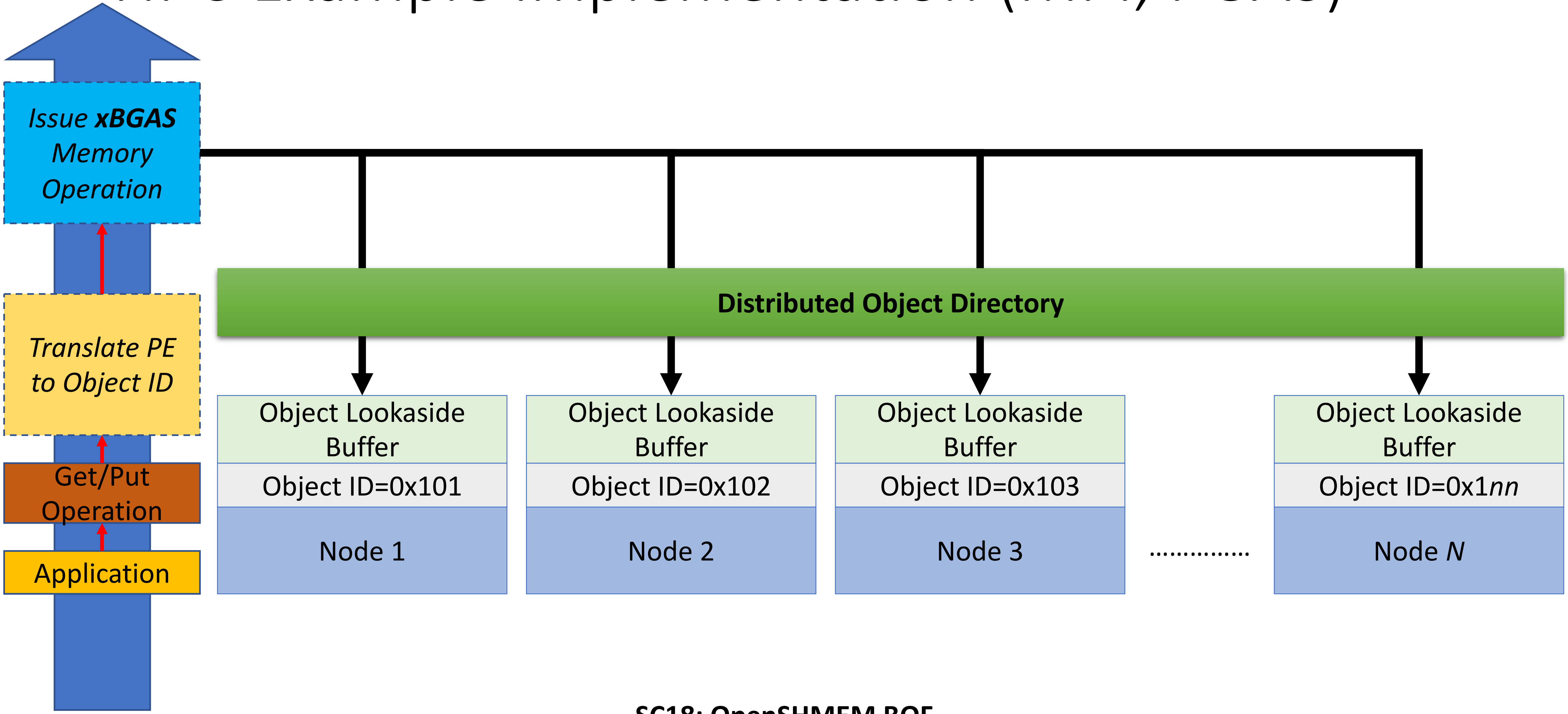
Assembly code from ***xbgas-asm-test***



```
sh zero,-62(s0)
sb zero,-63(s0)
ld a5,-24(s0)
eld a5,0(a5)
sd a5,-56(s0)
ld a5,-32(s0)
elw  a12,0(a12)
sw a5,-60(s0)
ld a5,-40(s0)
elh a5,0(a5)
sh a5,-62(s0)
ld a5,-48(s0)
elb a5,0(a5)
sb a5,-63(s0)
ld a5,-40(s0)
elhu a5,0(a5)
```

- *Up to 128 bits of address space*
- *Not necessarily contiguous!*
- *Most significant (extended) address can be object ID (as opposed to raw address)*

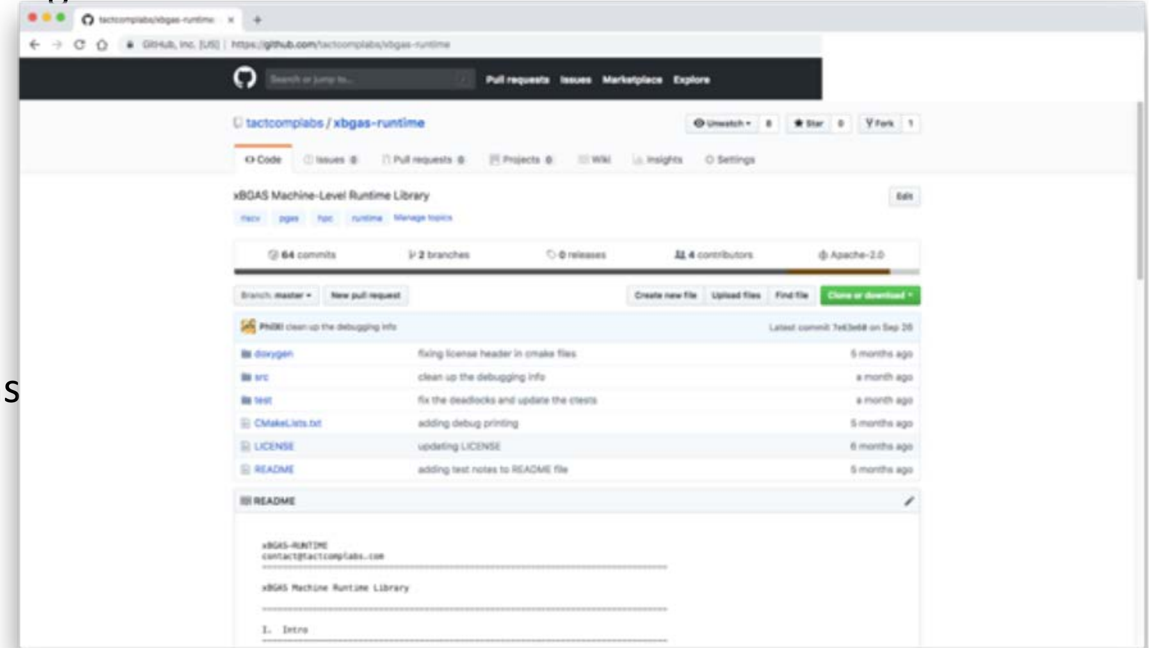
HPC Example Implementation (MPI, PGAS)



ongoing research

Research & Progress

- Software
  - Data Intensive Scalable Computing Lab at Texas Tech is leading the software research
  - Current SV128 ISA spec implemented in GCC and LLVM
  - RISC-V Spike simulators are available (SST simulator in progress)
  - **Runtime library in place to mimic OpenSHMEM**
- Hardware
  - TCL/LBNL/MIT leading hardware effort
  - Exploring pipelined and accelerator-based implementations
  - Pipelined implementation has begun in Freechips Rocket
- Security
  - Micron is leading the efforts on defining the memory and security models
- Other Topics
  - Operating system (context save info)
  - Debugging
  - Programming Model



xBGAS Machine-Level Runtime Library

clean up the debugging info

doxygen	fixing license header in cmake files	5 months ago
src	clean up the debugging info	a month ago
test	fix the deadlocks and update the ctests	a month ago
CMakeLists.txt	adding debug printing	5 months ago
LICENSE	updating LICENSE	6 months ago
README	adding test notes to README file	5 months ago

```
xBGAS-RUNTIME
contact@tactcomplabs.com
------------------------------------------------

xBGAS Machine Runtime Library

------------------------------------------------

1.  Intro
------------------------------------------------
```

SV128 Technical Work Group: RISC-V Foundation

- Call for participation:
  - RISC-V SV128 Technical Working Group
    - Chair: Steve Wallach

- Draft SV128 specification released at SC18
  - Memory model
  - Instruction set extension

- Paper published at SC18 Memory Centric HPC
  - *xBGAS: Toward a RISC-V ISA Extension for Global, Scalable Shared Memory*

Acknowledgments

- David Donofrio/Farzad Fatollahi-Fard: LBNL
- Kurt Keville: MIT
- Xi Wang, Brody Williams, Yong Chen: Texas Tech
- Frank Conlon: TactCompLabs/Texas Tech

TEXAS TECH UNIVERSITY

TCL Tactical Computing Labs

Massachusetts Institute of Technology

BERKELEY LAB

SV128