# MMUKO-OS RingBoot System

### OBINEXUS NSIGII-Verified Bootloader

A quantum-inspired operating system bootloader implementing stateless ring boot with non-deterministic finite automaton verification.

## Overview

MMUKO-OS (M for Mike, U for Uniform, K for Kilo, O for Oscar) is a bootloader that uses:

- **NSIGII Protocol**: Trinary verification (YES/NO/MAYBE)
- **Quantum State Model**: 8 qubits with half-spin (π/4) allocation
- **Ring Boot State Machine**: Sparse → Remember → Active → Verify
- **RIFT Header**: 8-byte verification header with checksum

## Project Structure

```
mmuko-os/
├── img/
│   └── mmuko-os.img        # 512-byte bootable image
├── include/
│   └── obiboot.h           # Boot system header
├── src/
│   └── obiboot.c           # Boot implementation
├── build.sh                # Build script
├── ringboot.sh             # VirtualBox test script
└── README.md               # This file
```

## Core Concepts

### NSIGII Verification States

c

```c
#define NSIGII_YES   0x55  // Boot verified
#define NSIGII_NO    0xAA  // Boot failed
#define NSIGII_MAYBE 0x00  // Pending verification
```

### Quantum Spin Model

The system uses 8 qubits representing compass directions with half-spin allocation:

- **North** (0°): Starting position

- **Northeast** (π/4): First half-spin
- **East** (π/2): Quarter rotation
- **Southeast** (3π/4): Three-quarter point
- **South** (π): Opposite verification
- **Southwest** (5π/4): Return path
- **West** (3π/2): Final quadrant
- **Northwest** (7π/4): Completion check

### Boot State Machine

```
SPARSE → REMEMBER → ACTIVE → VERIFY
  ↓        ↓         ↓        ↓
Half    Memory    Full     NSIGII
Spin    Alloc     Process  Check
```

## Building

### Prerequisites

- GCC compiler
- NASM assembler
- VirtualBox (for testing)
- Bash shell

### Build Process

```bash
# Make scripts executable
chmod +x build.sh ringboot.sh

# Build boot image
./build.sh
```

Expected output:

```
==== MMUKO-OS Build System ====
[1/5] Compiling obiboot.c...
[2/5] Running NSIGII verification test...
✓ NSIGII verification PASSED
[3/5] Creating bootable boot sector...
[4/5] Assembling boot sector...
[5/5] Verifying boot image...
✓ Boot image is exactly 512 bytes
✓ Boot signature (0x55AA) verified
```

## Testing in VirtualBox

### Automated Setup

```
bash

./ringboot.sh
```

This script will:

1. Create a new VirtualBox VM
2. Attach the boot image as a floppy disk
3. Configure serial output logging
4. Start the VM
5. Monitor the boot sequence

### Manual VirtualBox Setup

1. Create new VM:
    - Name: MMUKO-OS-RingBoot
    - Type: Other
    - Version: Other/Unknown
    - RAM: 64MB
2. Add floppy controller:
    - Settings → Storage → Add Floppy Controller
    - Attach `img/mmuko-os.img`
3. Configure boot order:
    - Settings → System → Boot Order
    - Enable only Floppy
4. Start VM

### Expected Boot Sequence

The VM should display:

```
OBINEXUS MMUKO-OS RINGBOOT
NSIGII VERIFIED: BOOT SUCCESS
```

Then halt with code `0x55` (NSIGII_YES).

## Technical Details

### RIFT Header Format

| Offset | Size | Field | Value | Description |
|---|---|---|---|---|
| 0x00 | 4 | Magic | "NXOB" | OBINEXUS signature |
| 0x04 | 1 | Version | 0x01 | Protocol version |
| 0x05 | 1 | Reserved | 0x00 | Reserved for future use |
| 0x06 | 1 | Checksum | 0xFE | XOR checksum of header |
| 0x07 | 1 | Flags | 0x01 | Boot flags |

### MUCO Boot Sequence

Based on your documents, the MUCO (M for Mike, U for Uniform, C for Charlie, O for Oscar) boot sequence follows this pattern:

1. **Initialization**: All 8 qubits face NORTH (0°)
2. **North/East Check**:
   - Qubit 0: NORTH (0°)
   - Qubit 1: NORTHEAST (π/4)
   - Qubit 2: EAST (π/2)
3. **Memory State**: Transition to REMEMBER state
4. **South/West Verification**:
   - Qubit 4: SOUTH (π)
   - Qubit 5: SOUTHWEST (5π/4)
   - Qubit 6: WEST (3π/2)
5. **NSIGII Verification**: Check if 6+ qubits are properly aligned
6. **HALT**: Store verification code (0x55 for success) in AL register

### Half-Spin Allocation

The system uses **half-spin** (π/4 rotations) to implement:

- **Double space, half time**: When in SPARSE state
- **Half space, double time**: When in ACTIVE state
- **Auxiliary star sequences**: No noise/noise, stop/start patterns

This corresponds to the polar coordinate model described in your documents where:

- Each half-spin represents π/4 radians (45°)
- Full rotation is 8 half-spins (2π radians)
- State preservation uses conjugate pairs

### Non-Deterministic State Machine

The Ring Boot state machine implements NDFA logic:

- **Sparse State**: System has memory allocated but is inactive
- **Remember State**: Previous state is preserved for verification
- **Active State**: Full processing with all qubits synchronized
- **Verify State**: NSIGII protocol checks state transitions

Transitions are **non-deterministic** in that the same input (qubit direction) can yield different verification results based on:

- Transition count
- Previous state
- Number of verified qubits

## Verification Protocol

### NSIGII (Nigeria Signature II) Logic

The verification uses **trinary logic**:

```c
if (verified_count >= 6) return NSIGII_YES;   // 0x55
if (verified_count < 3)  return NSIGII_NO;    // 0xAA
else                     return NSIGII_MAYBE; // 0x00
```

This implements the **three-pointer system** from your documents:

- **Type**: What kind of state? (SPARSE/ACTIVE/REMEMBER)
- **Value**: What is the current value? (0-7 for spin direction)
- **Memory**: What memory is allocated? (half-spin allocation)

## Troubleshooting

### Boot image not 512 bytes

If `build.sh` reports incorrect size:

```bash
# Check assembly output
cat build/obiboot_sector.asm
```

Ensure padding is correct:

```nasm
times 510-($-$$) db 0
dw 0xAA55
```

### VM doesn't boot

1. Check boot order in VirtualBox settings
2. Verify floppy controller is attached
3. Ensure boot signature is present:

```bash
xxd -s 510 -l 2 img/mmuko-os.img
# Should show: 55aa
```

### NSIGII verification fails

If VM displays "NSIGII FAILED: HALT":

1. Check qubit initialization in `obiboot.c`
2. Verify half-spin allocation logic
3. Ensure transition count is correct

## Future Enhancements

- [ ] Full 8-qubit verification with all compass directions

- [ ] Lambda integration for energy measurement
- [ ] Tomographic onion layer encryption
- [ ] Multi-stage RIFT pipeline (TOKENISER → PARSER → AST)
- [ ] GossiLang coroutine bindings for PHP/Python/Node
- [ ] NLINK automaton state minimization

## References

From your documentation:

- **NSIGII Protocol**: Symbolic interpretation with pointer-based intent resolution
- **MUCO Boot**: Auxiliary star sequences with no-noise/noise patterns
- **Ring Boot**: Stateless active/sparse state machine with memory preservation
- **RIFT Ecosystem**: Single-pass compilation with policy-based thread safety

## License

Part of the OBINEXUS Computing Framework by Nnamdi Michael Okpala.

## Contact

For questions about NSIGII verification or MUCO boot sequences, refer to the source documents or the RIFT Ecosystem documentation.

---

**Built with care, compiled with intent, verified with NSIGII.**