

# Shakti RISC-V Setup Guide

MacBook M4 / Apple Silicon • ACAD Course

## Shakti RISC-V: Complete Setup Guide (Zero to Lab-Ready)

### MacBook M4 / Apple Silicon • ACAD Course

**Version:** 1.0

**Author:** StarkAg

**Repository:** <https://github.com/StarkAg/shakti-workspace>

---

## Overview

This guide takes you from a fresh Mac M4 (no VM, no tools) to a fully working Shakti RISC-V development environment capable of building programs and flashing the Arty A7 at the lab.

**Why this guide?** The official ACAD setup uses VirtualBox + Ubuntu. VirtualBox does **not** run on Apple Silicon (M1/M2/M3/M4). This guide uses **UTM** (free, native) + **Ubuntu ARM64** instead.

**Total time:** ~1–2 hours (mostly unattended: VM install + toolchain build)

---

## Table of Contents

1. [Prerequisites](#)
  2. [Part 1: Install UTM](#)
  3. [Part 2: Download Ubuntu](#)
  4. [Part 3: Create and Install Ubuntu VM](#)
  5. [Part 4: Install System Packages](#)
  6. [Part 5: Build RISC-V Toolchain](#)
  7. [Part 6: Clone the Workspace](#)
  8. [Part 7: Build and Verify](#)
  9. [Part 8: Lab Flash \(Arty A7\)](#)
  10. [Quick Reference](#)
  11. [Troubleshooting](#)
-

## 1. Prerequisites

Item	Requirement
Mac	Apple Silicon (M1/M2/M3/M4)
RAM	8 GB minimum (6 GB for VM)
Disk	~50 GB free
Network	Internet for downloads
Lab hardware	Arty A7 FPGA (pre-flashed with Shakti bitstream)

### Why a “Fresh” FPGA Isn’t Enough

- A brand-new Arty A7 has **no FPGA design loaded**.
- The Shakti core (RISC-V processor) must first be **built and downloaded** using Vivado.
- Vivado is **x86 only** and heavy; it **doesn’t run on Mac/Apple Silicon**.
- So students usually don’t do this themselves—it’s done **once per board** by lab staff or course admins.

### What You Actually Need

- **Pre-flashed Arty A7:** Board already configured with the Shakti bitstream (lab boards are normally like this).

## 2. Part 1: Install UTM

UTM is a free virtualization app that runs natively on Apple Silicon.

**Option A: Download** - Go to <https://mac.getutm.app/> - Download and install UTM

**Option B: Homebrew**

```
brew install --cask utm
```

## 3. Part 2: Download Ubuntu

1. Go to <https://ubuntu.com/download/server/arm>
2. Download **Ubuntu 24.04 LTS Server (64-bit ARM)**
  - Or Desktop: <https://ubuntu.com/download/desktop> (choose ARM64)

Save the .iso file (e.g. ubuntu-24.04-live-server-arm64.iso).

## 4. Part 3: Create and Install Ubuntu VM

## 4.1 Create VM

1. Open **UTM**
2. Click **Create a New Virtual Machine**
3. Select **Virtualize** (important: not Emulate)
4. Select **Linux**
5. **Browse** and select the Ubuntu ISO
6. Click **Continue**

## 4.2 Configure Hardware

Setting	Value	Notes
Memory	6 GB	Minimum 4 GB
CPU cores	4	More = faster toolchain build
Disk size	40 GB	Minimum for toolchain + workspace

Click **Save**, choose a name (e.g. Ubuntu-Shakti), and save.

## 4.3 Install Ubuntu

1. Start the VM
2. Follow the Ubuntu installer:
  - Language: English
  - Keyboard: Your layout
  - Install type: Default
  - Network: Configure if needed (DHCP usually works)
  - Storage: Use entire disk
  - Profile: Create username and password
  - SSH: Optional (install if you want remote access)
3. Reboot when prompted
4. Log in with your credentials

# 5. Part 4: Install System Packages

Open a **terminal** inside the Ubuntu VM and run:

```
sudo apt update
sudo apt install -y build-essential git autoconf automake libtool
texinfo \
    flex bison libmpc-dev libmpfr-dev libgmp-dev gawk python3
python3-pip \
    openocd tmux
```

**What these do:** - build-essential – Compiler, make - autoconf, automake, libtool – Build system for toolchain - flex, bison – Parser generators - libmpc-dev, libmpfr-dev, libgmp-dev – Math libs for GCC - openocd – ARM64-native OpenOCD for flashing (required; shakti-tools OpenOCD is x86 only) - tmux – Optional; used by shakti.sh for debug sessions

## 6. Part 5: Build RISC-V Toolchain

The prebuilt shakti-tools are **x86\_64 only** and do not run on ARM64. You must build the toolchain.

### 6.1 Clone riscv-gnu-toolchain

```
cd ~
git clone https://github.com/riscv-collab/riscv-gnu-toolchain
cd riscv-gnu-toolchain
```

### 6.2 Configure and Build

```
./configure --prefix=$HOME/riscv32 --with-arch=rv32imac --with-abi=ilp32
make -j$(nproc)
```

**Time:** ~30–60 minutes (depends on CPU/RAM)

### 6.3 Verify

```
$HOME/riscv32/bin/riscv32-unknown-elf-gcc --version
```

You should see something like: riscv32-unknown-elf-gcc (GCC) 15.x.x

### 6.4 If You See libgcc.a Errors Later

If builds fail with libgcc.a not found:

```
# Find your GCC version
ls $HOME/riscv32/lib/gcc/riscv32-unknown-elf/

# Copy libgcc.a from shakti-tools if you have it (replace
YOUR_VERSION)
# cp /path/to/shakti_workspace/shakti-tools/riscv32/lib/gcc/riscv32-unknown-elf*/libgcc.a \
# $HOME/riscv32/lib/gcc/riscv32-unknown-elf/YOUR_VERSION/
```

---

## 7. Part 6: Clone the Workspace

The workspace includes the SDK, fixes, elf2hex, env.sh, and lab-flash script.

**Inside the Ubuntu VM:**

```
cd ~
git clone https://github.com/StarkAg/shakti-workspace.git
mv shakti-workspace shakti_workspace
```

**Workspace contents:** - shakti-sdk/ – Shakti SDK (GCC 15 compatible) - bin/elf2hex – ARM64-compatible ELF-to-hex tool - env.sh – Environment setup - lab-flash.sh – One-command lab flash - docs/ – Documentation

---

## 8. Part 7: Build and Verify

### 8.1 Source Environment

```
cd ~/shakti_workspace
source env.sh
```

Expected output: Shakti environment ready [self-built].

### 8.2 Build Hello Example

```
cd shakti-sdk
make software PROGRAM=hello TARGET=artix7_35t
```

Or for Parashu target:

```
make software PROGRAM=hello TARGET=parashu
```

### 8.3 Verify Output

The built ELF should be at:

shakti-sdk/software/examples/uart\_applns/hello/output/hello.shakti

If this succeeds, your setup is complete.

---

## 9. Part 8: Lab Flash (Arty A7)

When you are at the lab with the Arty A7:

### 9.1 Connect Hardware

1. Connect Arty A7 to your Mac via USB (JTAG)
2. In UTM: VM → **Removable Devices** → **[Digilent USB Device]** → **Connect**
3. In the VM, run: `lsusb` – you should see the Digilent device

### 9.2 Flash

```
cd ~/shakti_workspace
source env.sh
./lab-flash.sh hello artix7_35t
```

You will be prompted for `sudo` (OpenOCD needs root for USB).

### 9.3 View Output

- **UART:** 115200 baud (e.g. `screen /dev/ttyUSB1 115200`, PuTTY, or minicom)
  - You should see: Hello world !
-

## 10. Quick Reference

Task	Command
Setup environment	<code>source env.sh</code>
Build hello (Arty A7)	<code>make software PROGRAM=hello TARGET=artix7_35t</code>
Build hello (Parashu)	<code>make software PROGRAM=hello TARGET=parashu</code>
List examples	<code>make list_applns</code>
List targets	<code>make list_targets</code>
Flash at lab	<code>./lab-flash.sh hello artix7_35t</code>
UART terminal	<code>screen /dev/ttyUSB1 115200</code>

## 11. Troubleshooting

### “Exec format error” (elf2hex or openocd)

You are using x86\_64 binaries. Use this workspace’s bin/elf2hex and system OpenOCD. Run `source env.sh` before building.

### “riscv32-unknown-elf-gcc: command not found”

Toolchain not in PATH. Run `source env.sh` from the workspace root.

### Build fails with CSR / undefined reference errors

The workspace includes the needed fixes. If using vanilla SDK, update Makefiles: `rv32imac` → `rv32imac_zicr`.

### USB device not visible in VM

- UTM → VM → Removable Devices → Connect Digilent USB
- Disconnect from host first if needed

### OpenOCD: “libusb\_open() failed”

- Confirm USB is passed through to the VM
- Run with `sudo` (e.g. `sudo openocd ...`)

### Toolchain build runs out of memory

- Reduce parallelism: `make -j2` instead of `make -j$(nproc)`
  - Give the VM more RAM (6–8 GB)
-

## Summary Checklist

- ☐ UTM installed
  - ☐ Ubuntu 24.04 ARM64 VM created and installed
  - ☐ System packages installed (`apt install build-essential openocd ...`)
  - ☐ RISC-V toolchain built (`$HOME/riscv32`)
  - ☐ Workspace cloned (`git clone https://github.com/StarkAg/shakti-workspace.git`)
  - ☐ `source env.sh` and `make software PROGRAM=hello TARGET=artix7_35t` succeeds
  - ☐ At lab: Arty A7 connected, USB passthrough, `./lab-flash.sh hello artix7_35t`
- 

**Good luck with your ACAD lab!**

For issues or updates: <https://github.com/StarkAg/shakti-workspace>