



SPIKE your Code: A RISC-V ISA Workshop

Maktab-e-Digital Systems (MEDS)
UET Lahore

July 28, 2025

Workshop Agenda:

Session 1: Why RISC-V ISA Layer? Recall RV32I Basics

Session 2: Demo Code

----- Task 1 -----

Session 3: Control Flow and Pseudo Ops

----- Task 2 -----

Session 4: Spike Simulation & Logging

----- Task 3 -----

Session 5: Optimization Strategies

----- Task 4 -----

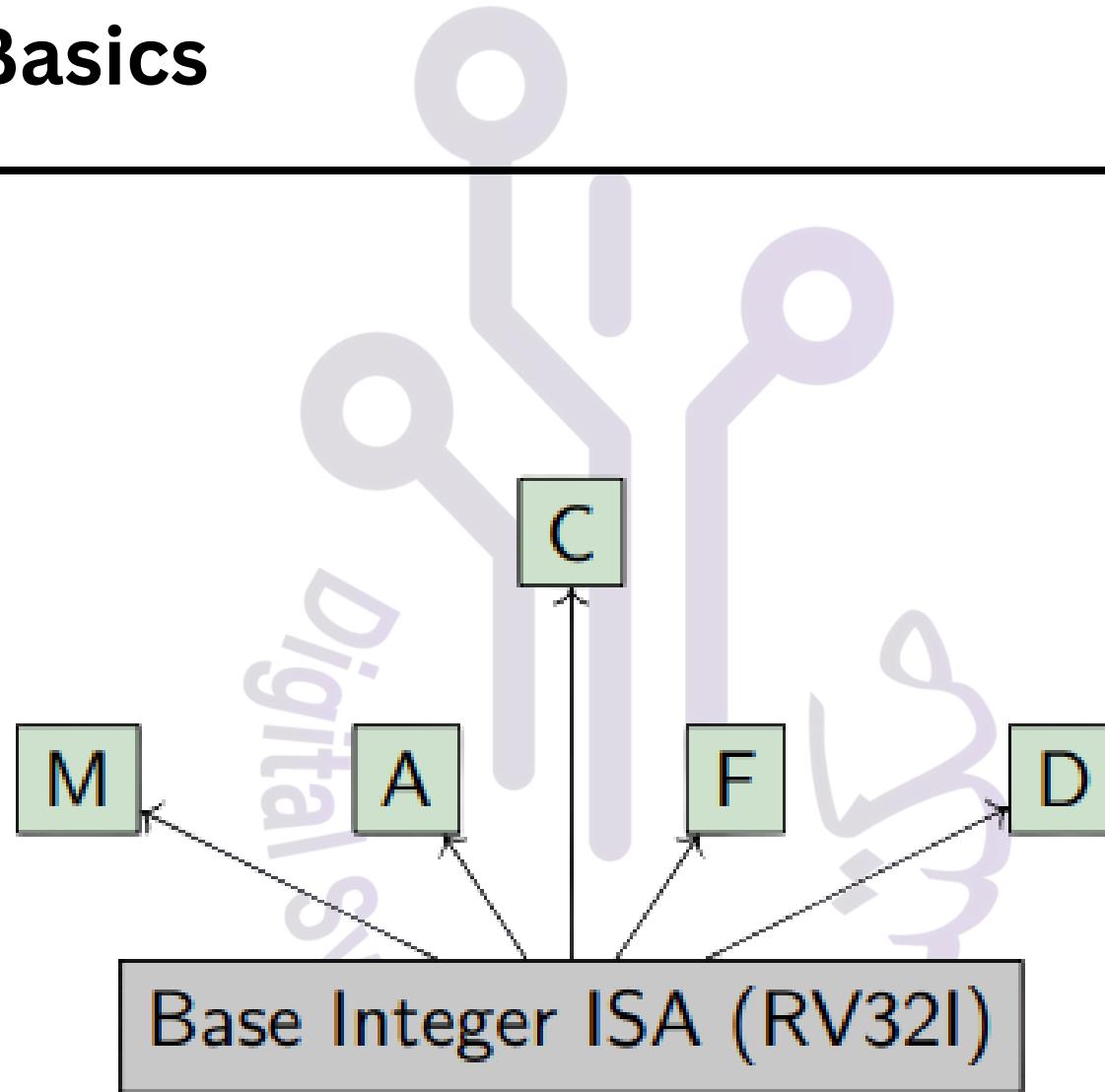
Why RISC-V ISA?

- Hardware-Software Boundary
- Microarchitectural Design
- Verification & Compliance Testing
- Portability & Compatibility

The logo for MEDS (Microelectronic Design Systems) features the letters "MEDS" in a large, bold, black sans-serif font. Behind the letters is a faint watermark consisting of a stylized circuit board or gear design with the words "Digital Systems" written diagonally across it.

MEDS

RV32I Basics

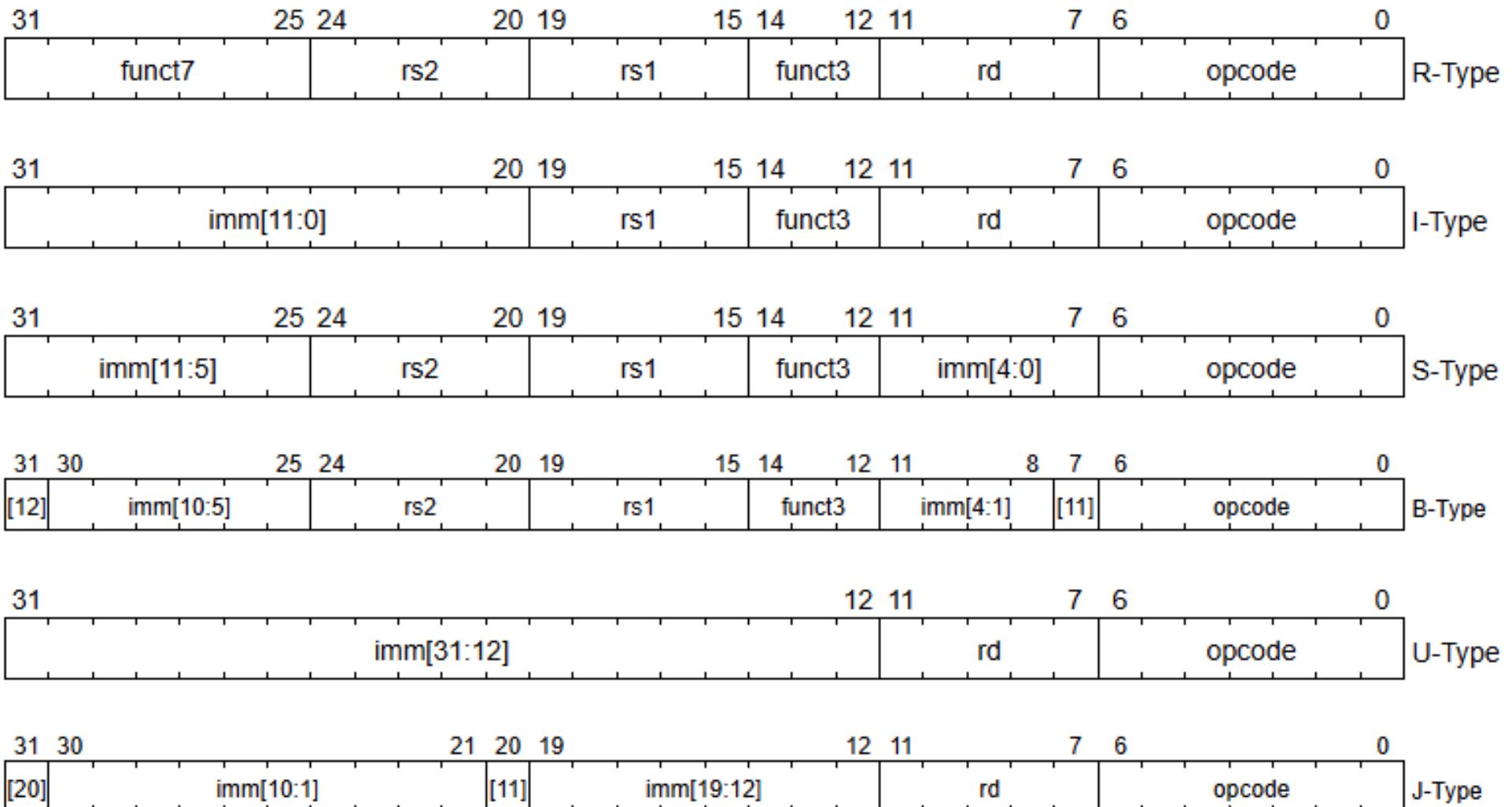


RV32I Base Integer ISA

RV32I Register File

| Register name | Symbolic name | Description | Saved by |
|-----------------------------|---------------|--------------------------------------|----------|
| 32 integer registers | | | |
| x0 | Zero | Always zero | |
| x1 | ra | Return address | Caller |
| x2 | sp | Stack pointer | Callee |
| x3 | gp | Global pointer | |
| x4 | tp | Thread pointer | |
| x5 | t0 | Temporary / alternate return address | Caller |
| x6–7 | t1–2 | Temporary | Caller |
| x8 | s0/fp | Saved register / frame pointer | Callee |
| x9 | s1 | Saved register | Callee |
| x10–11 | a0–1 | Function argument / return value | Caller |
| x12–17 | a2–7 | Function argument | Caller |
| x18–27 | s2–11 | Saved register | Callee |
| x28–31 | t3–6 | Temporary | Caller |

RV32I Instruction Format



Subset of RV32I Instructions

Arithmetic & Logic (R-type):

- add rd, rs1, rs2
- sub rd, rs1, rs2
- and rd, rs1, rs2
- or rd, rs1, rs2
- xor rd, rs1, rs2
- slt rd, rs1, rs2

Immediate (I-type):

- addi rd, rs1, imm
- andi rd, rs1, imm
- ori rd, rs1, imm
- xori rd, rs1, imm
- slti rd, rs1, imm

Memory (I/S-type):

- lw rd, offset(rs1)
- sw rs2, offset(rs1)

Branch (B-type):

- beq rs1, rs2, offset
- bne rs1, rs2, offset
- blt rs1, rs2, offset
- bge rs1, rs2, offset

Jump (J-type):

- jal rd, offset

Demo Code

```
.section .text
```

```
.globl _start
```

```
_start:
```

```
    li a0, 7
```

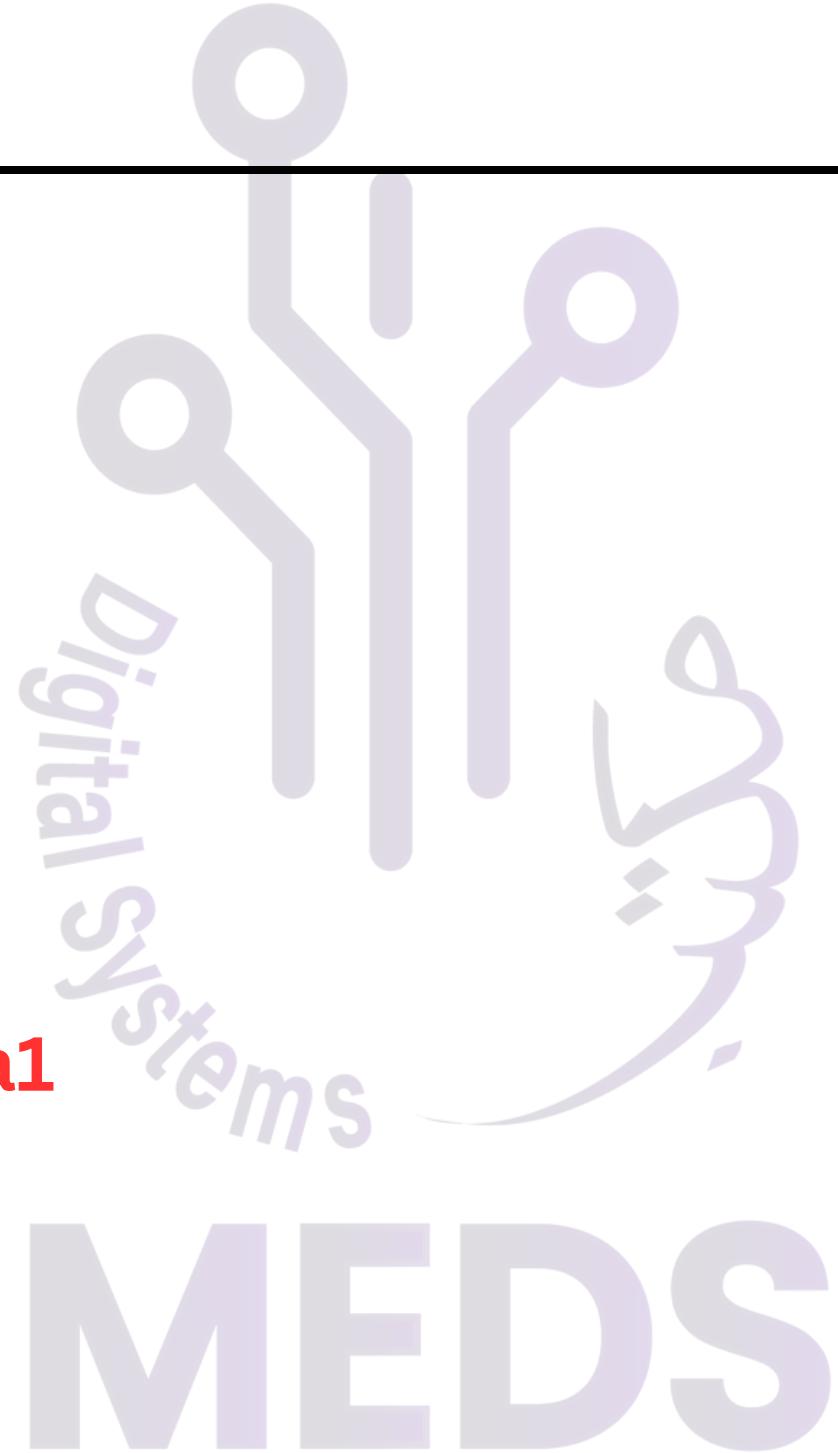
```
    li a1, 5
```

```
    add a2, a0, a1
```

```
    mv a0, a2
```

```
    li a7, 93
```

```
    ecall
```

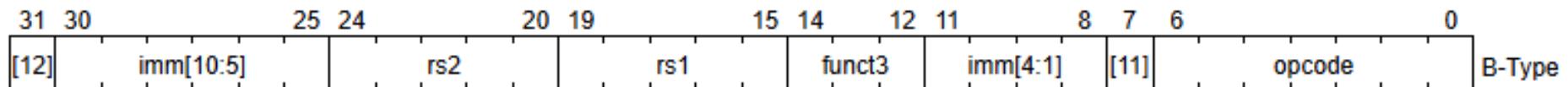


Task 1: Convert C to Assembly and Simulate

```
int sum(int *arr, int n) {  
    int total = 0;  
    for (int i = 0; i < n; i++)  
        total += arr[i];  
    return total;  
}  
  
int main() {  
    int data[] = {1,2,3,4,5};  
    return sum(data, 5);  
}
```

Control Flow and Pseudo Ops

Branch Instructions:



loop:

lw t0, 0(sp) # load element

beq t0, zero, done # if element == 0, exit loop

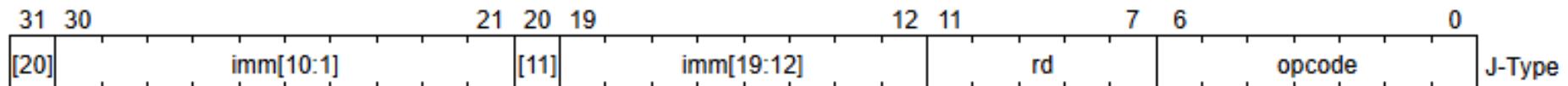
addi sp, sp, 4 # advance pointer

jal zero, loop # unconditional jump back

done:

Control Flow and Pseudo Ops

Jump Instructions:



Pseudo Instructions:

- li rd, imm
- mv rd, rs
- la rd, symbol
- j label
- ret
- nop

Task 2: Debug the code and Correct it

```
.section .text
.globl _start
_start:
    li t0, 0
    li t1, 10
loop:
    beq t0, t1, done  # should stop when t0 == 10?
    lw t2, 0(t0)      # wrong base register
    add t3, t3, t2    # t3 uninitialized
    addi t0, t0, 1
    jal zero, loop    # inefficient jump
done:
    mv a0, t3
    li a7, 93
    ecall
```

What is Spike?

Spike is the “golden” RISC-V ISA-simulator maintained by the RISC-V Foundation. It implements the full user-mode (and optionally privileged-mode) ISA in software, letting you:

- Validate your RTL against a reference model.
- Experiment with different ISA profiles (RV32I vs. RV64GC, plus extensions).
- Collect functional correctness, instruction counts, and simple performance metrics before you have silicon.

Spike Pk vs Raw Mode

Pk

- Proxy-Kernel
- spike pk code
- Runs under a tiny proxy kernel that provides Unix style sys calls
- General User Programs that require I/O

Raw

- spike --isa=rv32i code
- Executes the ELF image directly at address 0x80000000 (no syscalls, no loader).
- Bare-metal firmware development

Selecting ISA

--isa=<profile>

Examples:

--isa=rv32i

--isa=rv32imac (I + M + A + C extensions)

--isa=rv64gc (64-bit, G standard extensions)

This controls which instruction encodings Spike will accept and how it decodes.

MEDS

Instruction Loggings

--log=<file>

Tells Spike to append one line per retired instruction to <file>.

Each line contains:

PC of the instruction (hex)

Assembly mnemonic

Raw encoding (hex)

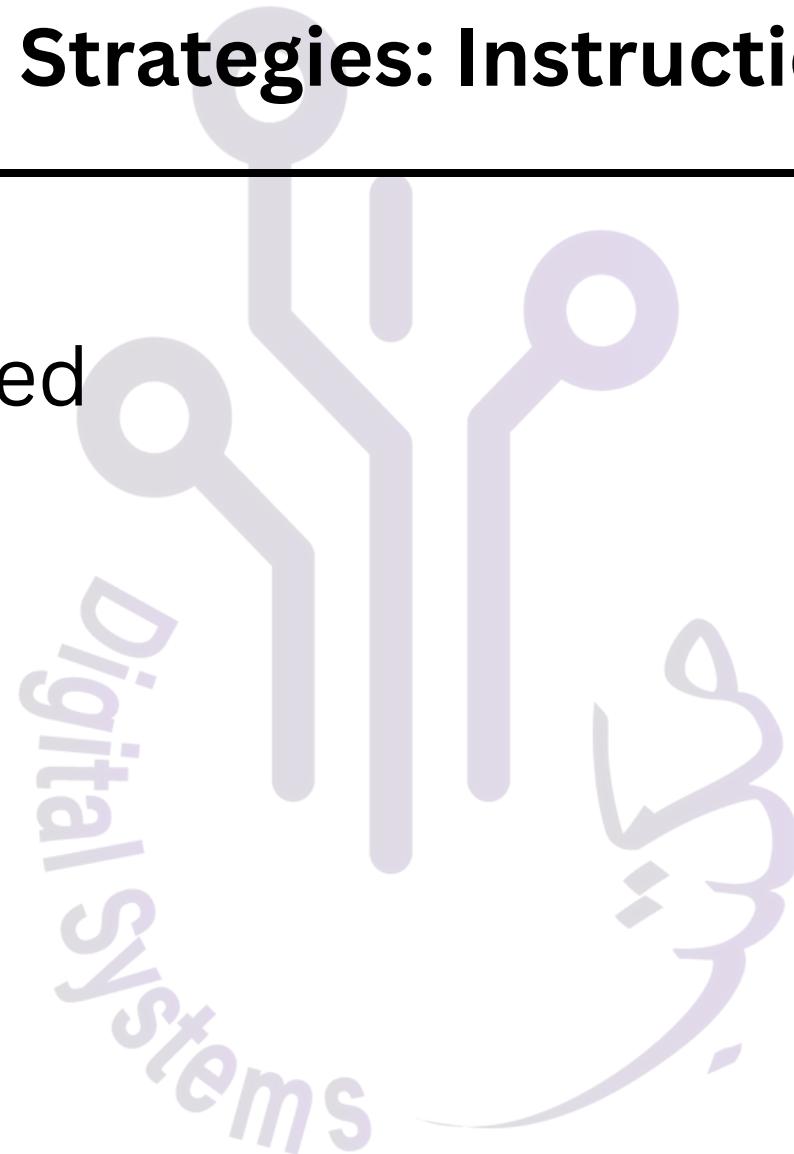
| | | |
|-------------|--------------|--------------|
| 0x00000000: | li a0,5 | # 0x00500513 |
| 0x00000004: | li a1,10 | # 0x00a00593 |
| 0x00000008: | add a2,a0,a1 | # 0x00b506b3 |
| 0x0000000c: | mv a0,a2 | # 0x00c50533 |
| 0x00000010: | li a7,93 | # 0x05d00713 |
| 0x00000014: | ecall | # 0x00000073 |

Task 3: Measure & Report Cycles

Measure the clock cycles of the corrected code of Task 2 and write a summary report showing which instruction is using how many clock cycles.

Optimization Strategies: Instruction Scheduling

```
# un-scheduled  
load t0,0(sp)  
addi t1,t2,1  
add t3,t0,t1
```



Optimization Strategies: Instruction Scheduling

```
# un-scheduled  
load t0,0(sp)  
addi t1,t2,1  
add t3,t0,t1
```

```
# rescheduled  
addi t1,t2,1 # independent  
load t0,0(sp)  
add t3,t0,t1
```

Optimization Strategies: Reducing Branches

Reducing Branch

Use set less than
Use add it

blt a0, a1, .inc # if x<y jump to .inc
j .done # else skip the increment

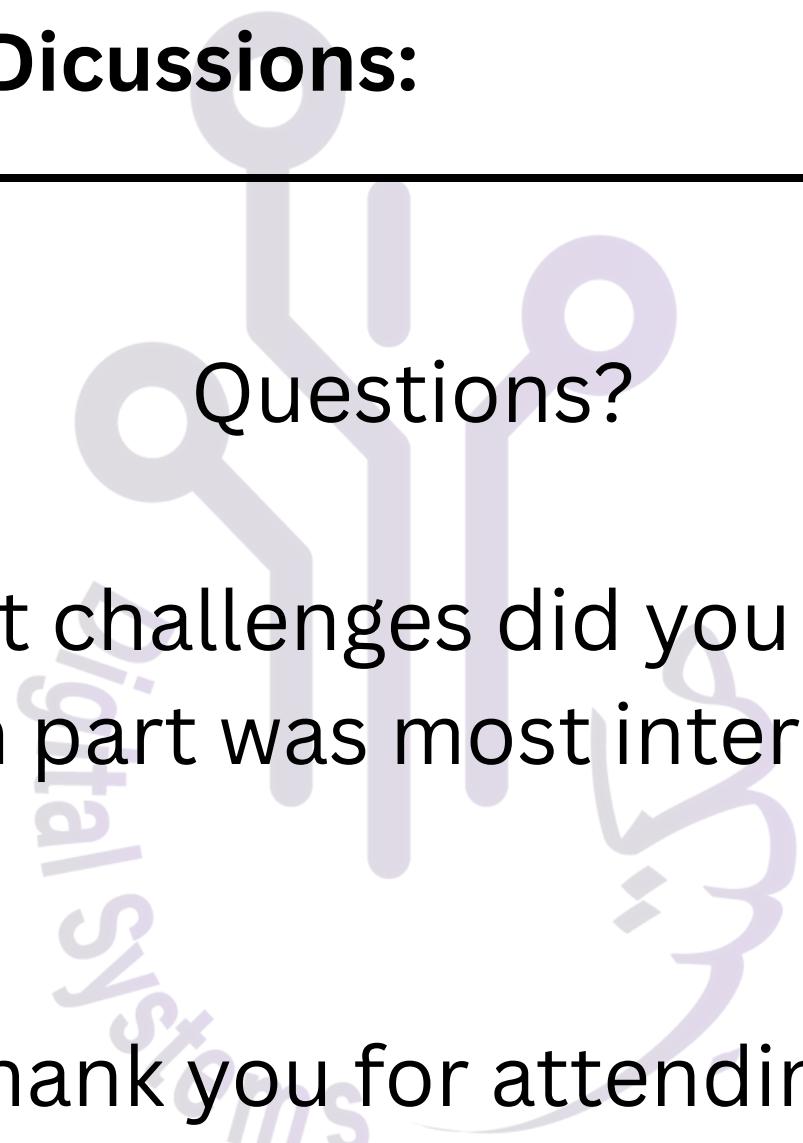
slt t0, a0, a1 # $t0 = (a0 < a1)?1:0$
add a2, a2, t0 # counter += t0

MEDS

Task 4: Code, Assemble, Spike, Document

Implement a function in C named as power(x,n) that takes two arguments x be the number and n be the power raise to that number. Convert this code in its assembly, assemble the code, compile the code, run the code on Spike, Check its cycle count and document which instructions are taking more cycles.

Questions & Discussions:



Questions?

What challenges did you face?
Which part was most interesting?

Thank you for attending!

MEDS