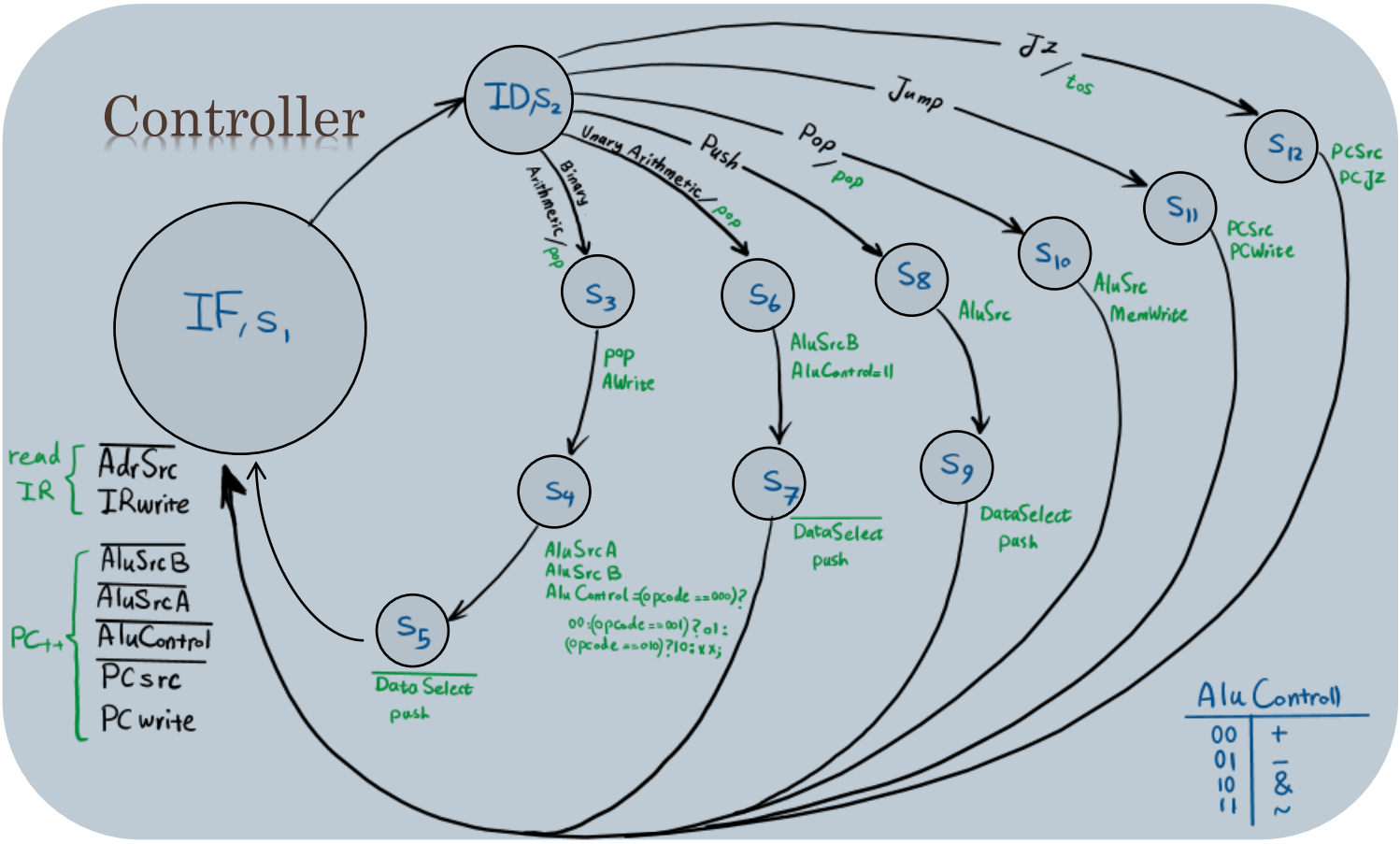
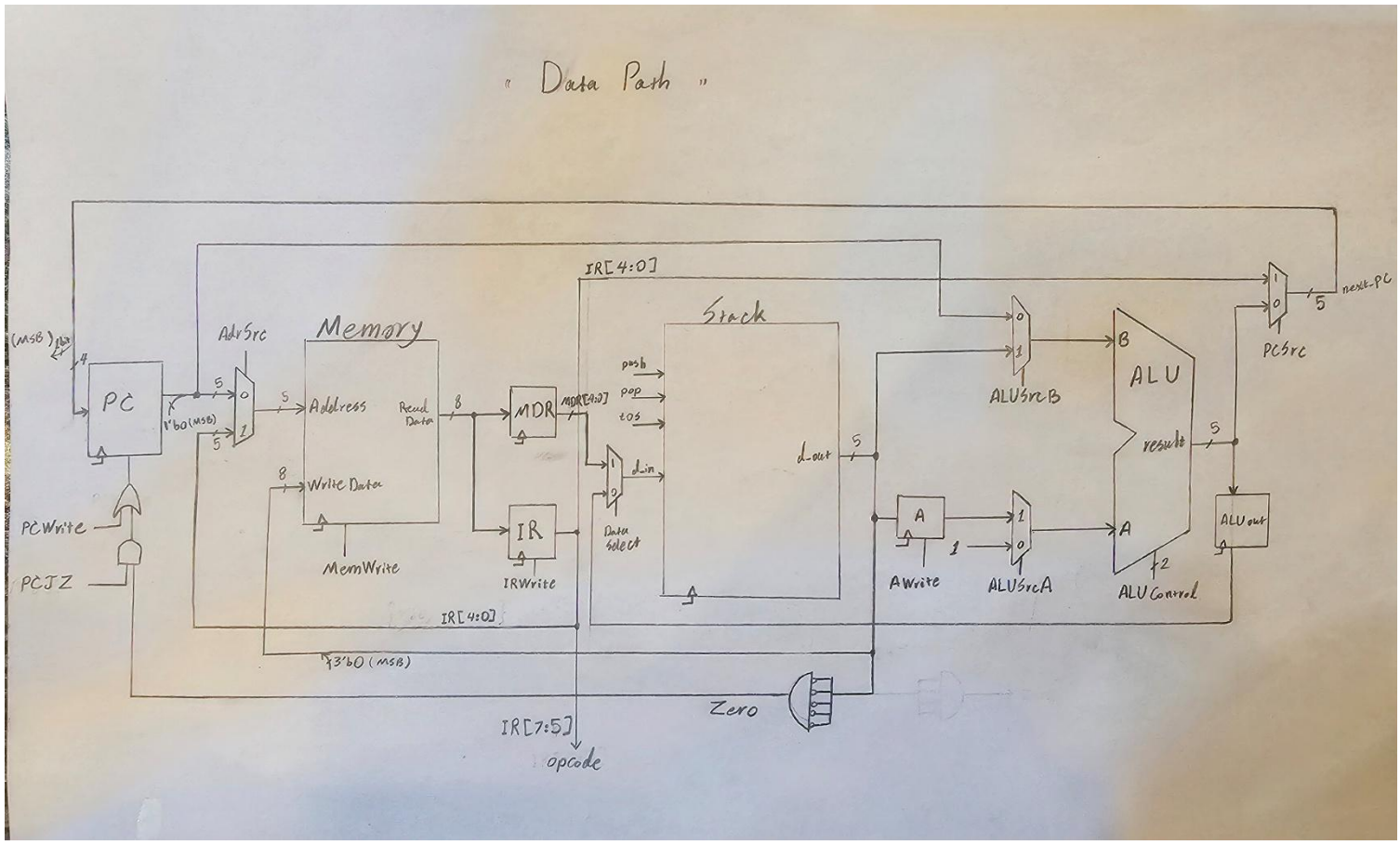
2025

multiCycle stack-based RiscV

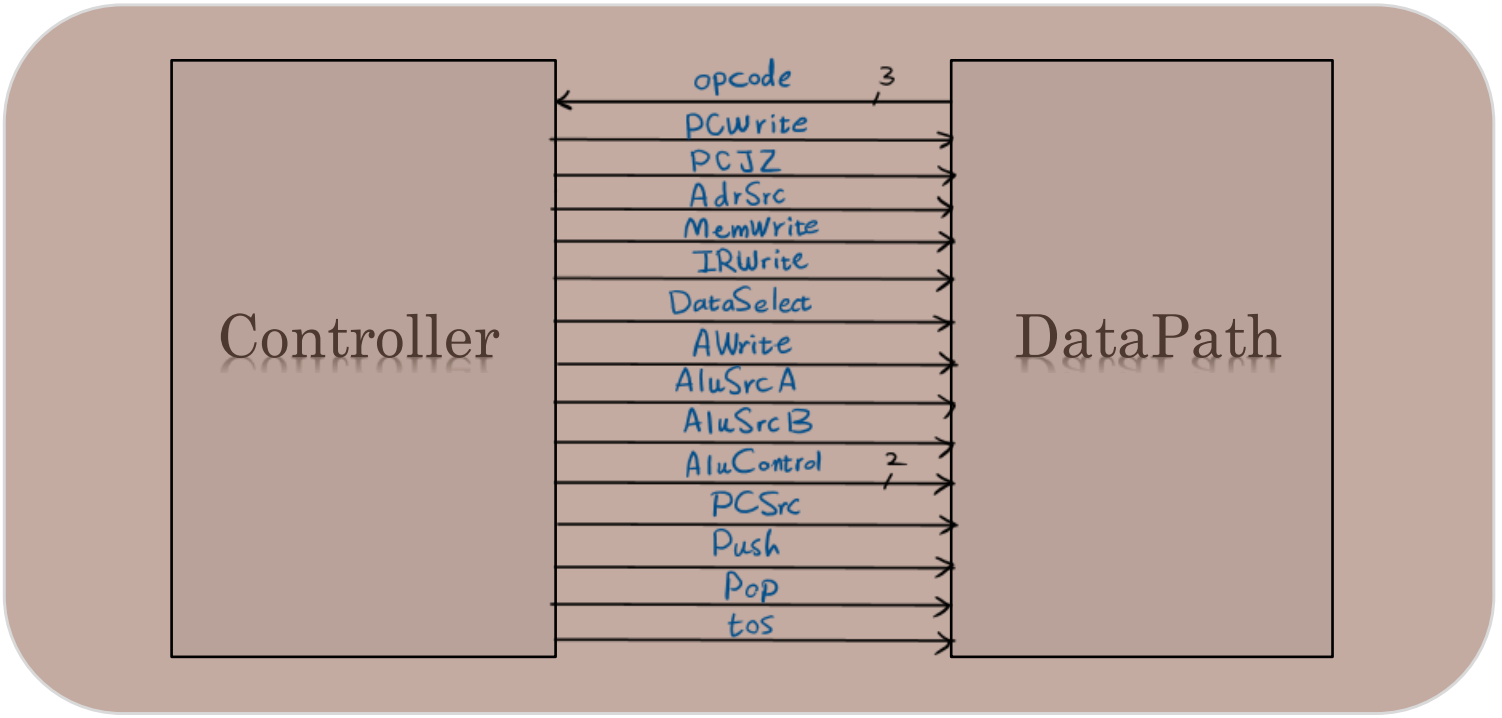
CONTRIBUTIONS
PARSA BUKANI & MOHAMMAD HOSSEIN MAZAHERI

TEHRAN UNIVERSITY


Controller
IF, S1
ID S2
Jz / tos
Jump
Pop / pop
Push
Unary Arithmetic / pop
Binary Arithmetic / pop
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
pop
AWrite
AluSrcB
AluControl=11
AluSrc
MemWrite
PCSrc
PCWrite
PCSrc
PCJz
read IR
AdrSrc
IRwrite
PC++
AluSrcB
AluSrcA
AluControl
PCsrc
PC write
AluSrcA
AluSrcB
Alu Control = (opcode == 000)?
00 : (opcode == 001) ? 01 :
(opcode == 010) ? 10 : xx;
Data Select
push
DataSelect
push
DataSelect
push
Alu Control
00 +
01 −
10 &
11 ~


" Data Path "
IR[4:0]
Memory
Stack
AdrSrc
PC
Address
Read Data
Write Data
MemWrite
MDR
IR
IRWrite
push
pop
tos
din
dout
Data Select
ALUSrcB
ALU
result
ALU out
A
AWrite
ALUSrcA
ALU Control
PCSrc
next-PC
PCWrite
PCJZ
IR[7:5]
opcode
Zero

Top Module



multiCycle Stack Based RiscV

- Instruction

Instruction Mnemonic and Definition		Bits 7:5
ADD	-	000
SUB	-	001
AND	-	010
NOT	-	011
PUSH	Load Address	100
POP	Store Address	101
JMP	Jump Address	110
JZ	Jump if Zero	111

دستورات این پردازنده به سه دسته تقسیم شده است:

دستورات محاسباتی منطقی:

- ADD : محتویات دو خانه‌ی بالای استک pop شده حاصل جمع آن‌ها مجددا روی استک push می‌شود.
- SUB : محتویات دو خانه‌ی بالای استک pop شده حاصل تفریق آن‌ها مجددا روی استک push می‌شود.
- AND : محتویات دو خانه‌ی بالای استک pop شده AND آن‌ها مجددا روی استک push می‌شود.
- NOT : محتویات خانه‌ی بالای استک pop شده NOT آن مجددا روی استک push می‌شود.

دستورات دسترسی به حافظه:

- PUSH : این دستور محتویات خانه‌ای از حافظه که توسط فیلد آدرس دستور مشخص شده است را روی استک push می‌کند.
- POP : این دستور محتویات خانه‌ی بالای استک را در خانه‌ای از حافظه که توسط فیلد آدرس دستور مشخص شده است pop می‌کند.

دستورات پرش:

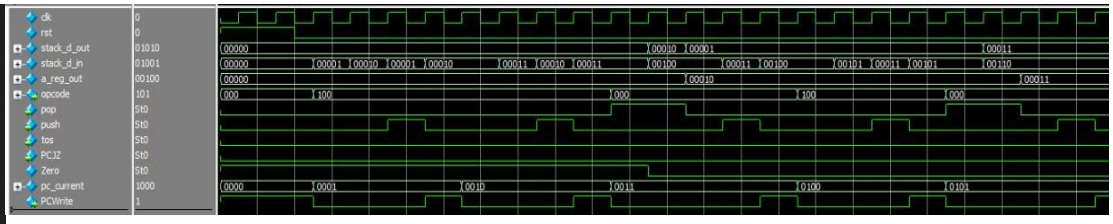
- JMP: این دستور به خانه‌ای از حافظه که توسط فیلد آدرس دستور مشخص شده است پرش می‌کند.
- JZ : این دستور در صورتی که محتویات بالای استک صفر باشد، به خانه‌ای از حافظه که توسط فیلد آدرس دستور مشخص شده است پرش می‌کند. توجه داشته باشید که این دستور محتویات بالای استک را pop نمی‌کند.

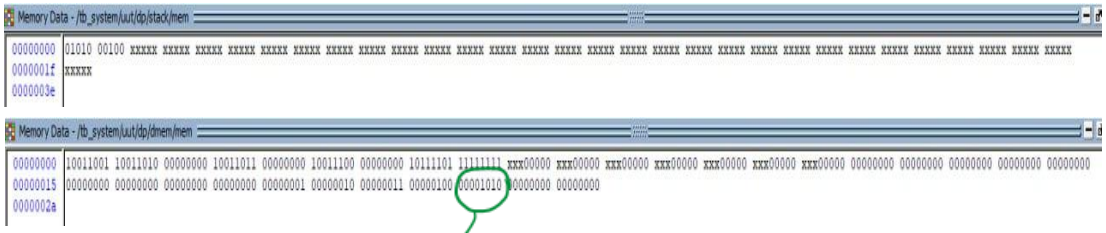
Simulation & Test

```
10011001   // [0] PUSH 25
10011010   // [1] PUSH 26
00000000   // [2] ADD
10011011   // [3] PUSH 27
00000000   // [4] ADD
10011100   // [5] PUSH 28
00000000   // [6] ADD
10111101   // [7] POP 29
11111111   // [8] HALT
xxx00000   // [9]  unused
xxx00000   // [10] unused
xxx00000   // [11] unused
xxx00000   // [12] unused
xxx00000   // [13] unused
xxx00000   // [14] unused
xxx00000   // [15] unused
00000000   // [16] unused
00000000   // [17] unused
00000000   // [18] unused
00000000   // [19] unused
00000000   // [20] unused
00000000   // [21] unused
00000000   // [22] unused
00000000   // [23] unused
00000000   // [24] unused
00000001   // [25] = 1
00000010   // [26] = 2
00000011   // [27] = 3
00000100   // [28] = 4
00000000   // [29] result = 10
00000000   // [30] unused
00000000   // [31] unused
```

Add 1+2+3+4

test Code





Result Register Add 1+2+3+4