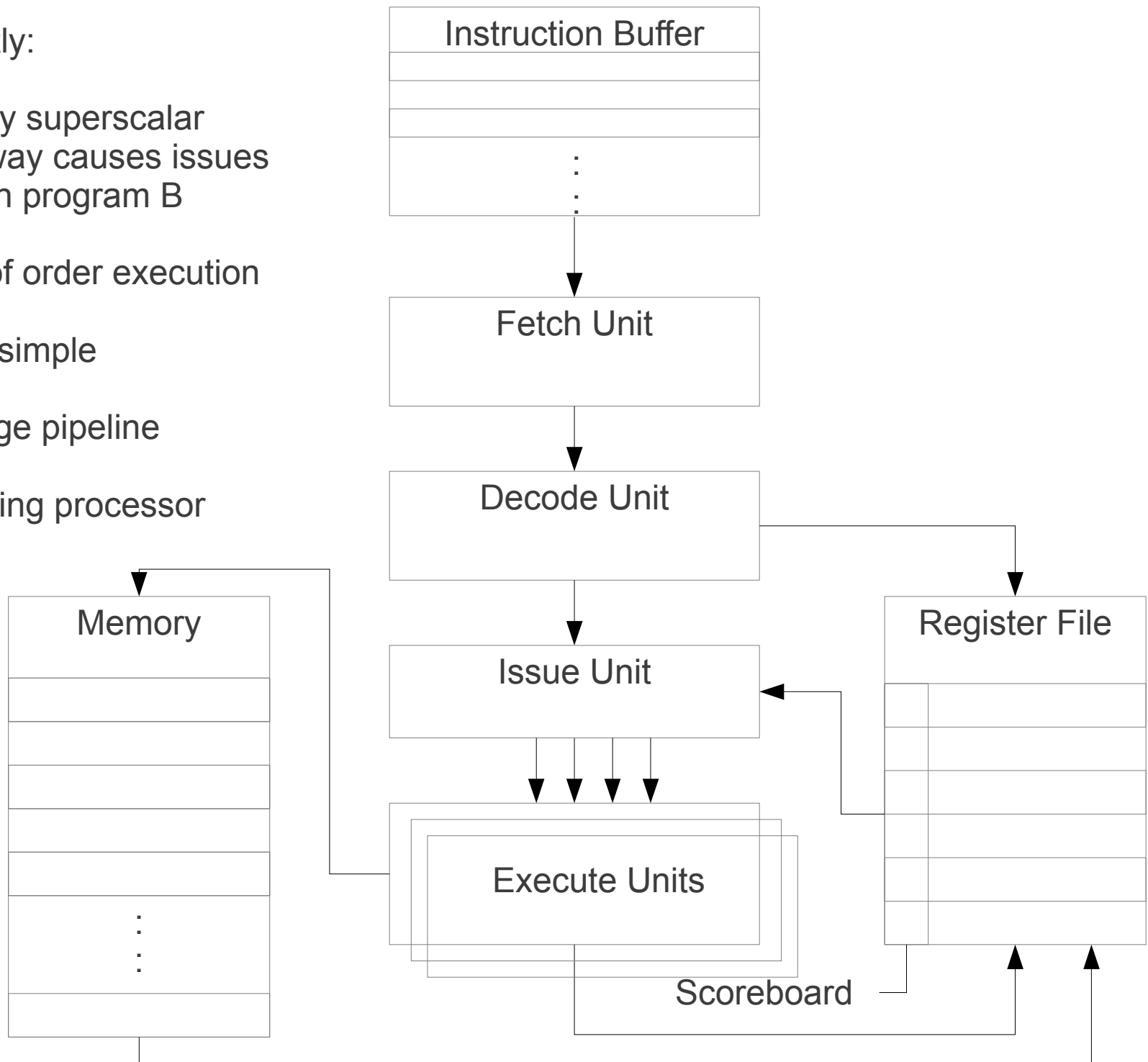
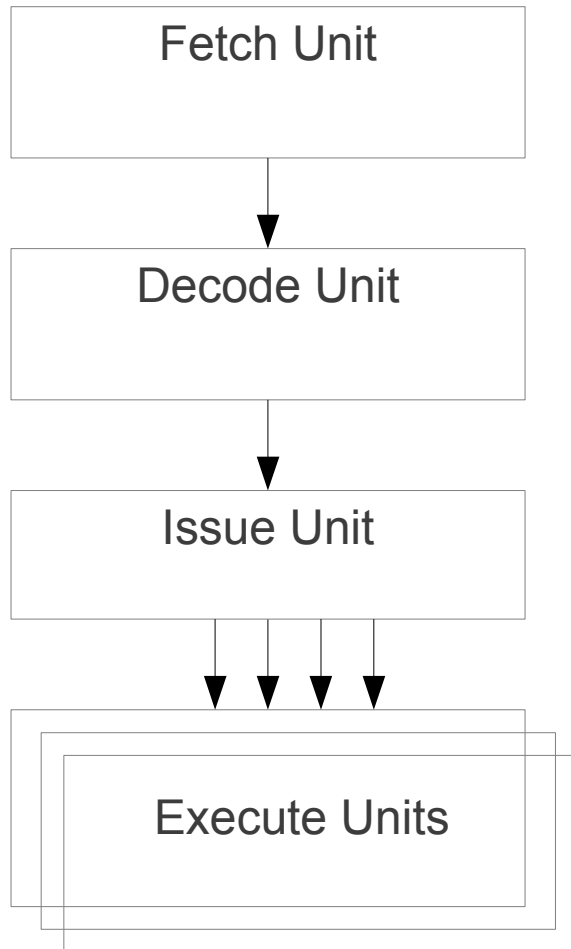


Currently:

- N-way superscalar
 - 2-way causes issues with program B
- Out of order execution
- Very simple
- 4 stage pipeline
- Working processor

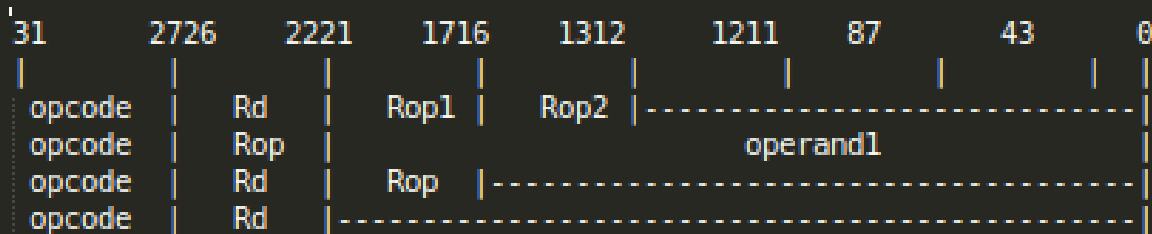


Features



- Fetches machine code
- Decodes the machine code, filling in information from the register block
- Issue the instructions if there are no dependencies
- Takes decoded instructions and execute them.

00000	ADD dest reg1 reg2
00001	SUB dest reg1 reg2
00010	MUL dest reg1 reg2
00011	DIV dest reg1 reg2
00100	CMP reg1 reg2
00101	MOV reg1 #
00110	LDR reg1 op1
00111	STR reg1 op1
01000	B reg1
01001	BLT reg1
01010	BE reg1
01011	BGT reg1
01110	END
01111	NOP



Mathematical operations (ADD, SUB, DIV, MUL)
 Data processing (MOV)
 Data movement (LDR, STR, CMP)
 Branch instructions (B, BLT, BE, BGT, JMP, RTN)

Experiments

a. Out of order Execution

```
mov r0, #0
mov r1, #1
mov r2, #1
mov r3, #51
loop: add r0, r0, r1
add r1, r1, r2
cmp r1, r3
blt .loop
end
```



0.7 instructions/clock cycle

1.5 instructions/clock cycle



```
mov r0, #0
mov r1, #1
mov r2, #1
mov r3, #51
loop: add r0, r0, r1
add r7, r7, r7
add r8, r8, r8
add r9, r9, r9
add r14, r14, r14
add r1, r1, r2
cmp r1, r3
sub r10, r10, r10
sub r11, r11, r11
add r7, r7, r7
add r8, r8, r8
add r9, r9, r9
add r14, r14, r14
blt .loop
add r7, r7, r7
add r8, r8, r8
add r9, r9, r9
end
```

Experiments

b. N-way scalar

S = 1

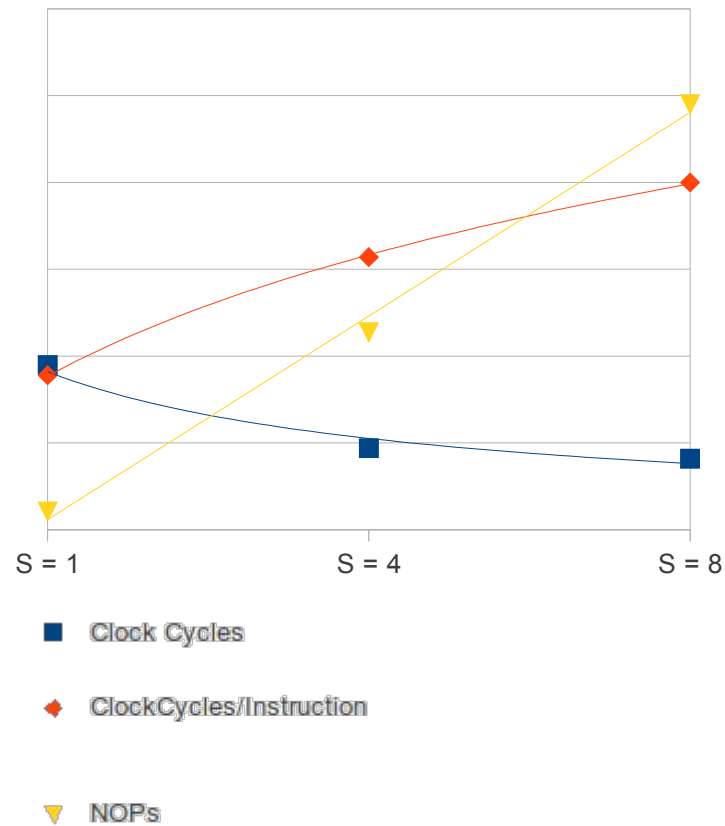
949 Clock Cycles
0.89 Instructions per cycle
106 NOPs

S = 4

470 Clock Cycles
1.57 Instructions per Cycle
1137 NOPs

S = 8

410 Clock Cycles
2.00 Instructions per Cycle
2451 NOPs



Experiments

c. Branch prediction

Due to the nature of the instruction set, having everything in registers, Branch prediction proved to be impossible.

However.

With the addition of Bi (and BLTi, BEi and BGTi) branch prediction become possible:

At $S = 4$:

A went from 0.7 to 1.8 I/CC

B went from 1.57 to 3.24 I/CC