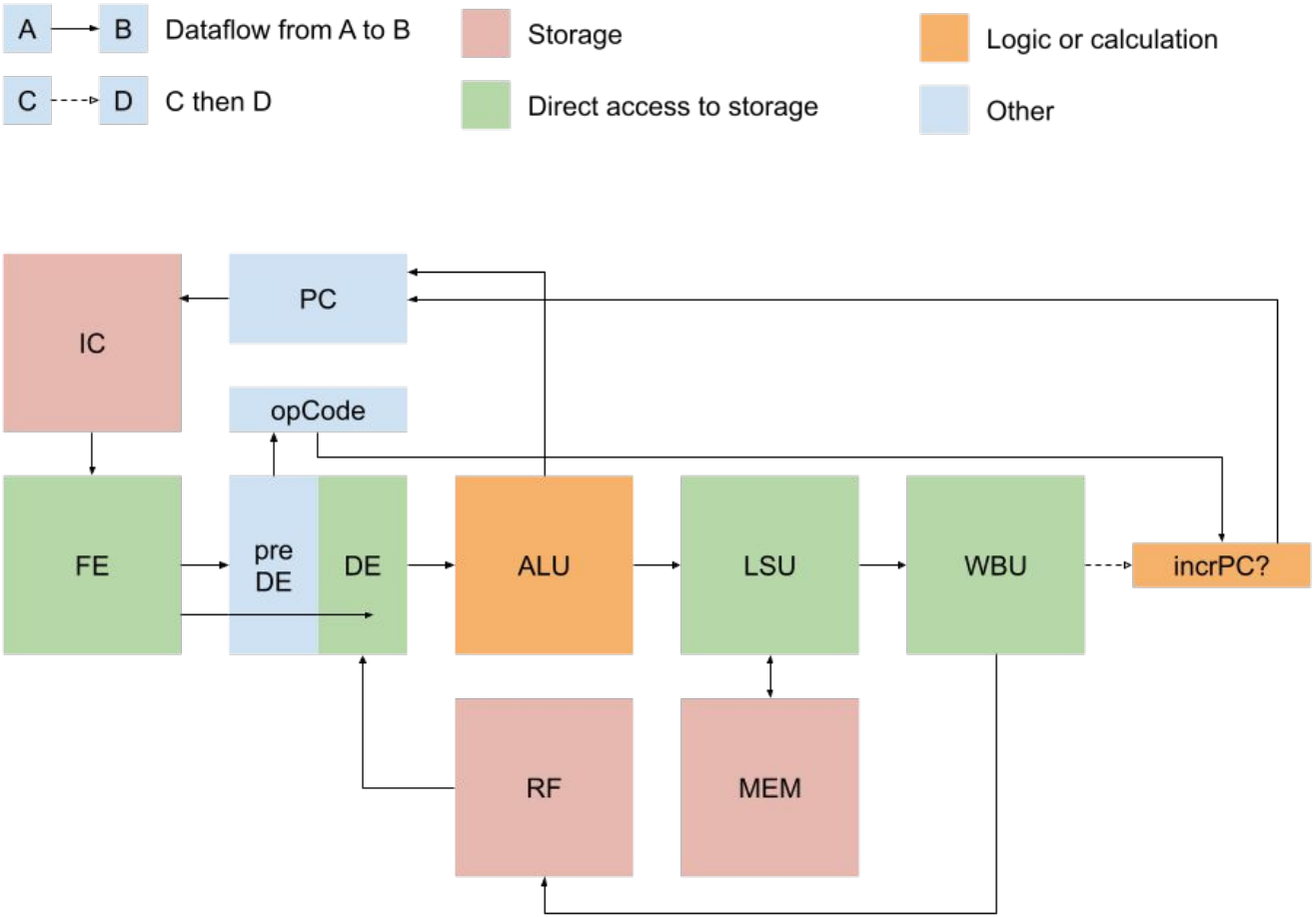
Processor Architecture



Instruction Set Architecture

add	rd	rs	rt	Reg[rd] = Reg[rs] + Reg[rt]
addi	rd	rs	#n	Reg[rd] = Reg[rs] + n
mul	rd	rs	rt	Reg[rd] = Reg[rs] + Reg[rt]
muli	rd	rs	#n	Reg[rd] = Reg[rs] + n
cmp	rd	rs	rt	Reg[rd] = rs < rs ? -1 : rs == rs ? 0 : 1
ld	rd	ro	#n	Reg[rd] = Mem[Reg[ro] + n]
ldc	rd	#n		Reg[rd] = Mem[n]
st	rs	ro	#n	Mem[Reg[ro] + n] = rf
brlz	rs	#n		rf <= 0 ? Pc = n : no-op
jplz	rs	#n		rf <= 0 ? Pc = Pc + n : no-op
br	#n			Pc = n
jp	#n			Pc = Pc + n

destination register ::= rd
source register ::= rs | rt
immediate ::= #n

registers

```
zero, //constant zero
ra, //return address
sp, //stack pointer
gp, //global pointer
tp, //thread pointer
t0, t1, t2, t3, t4, t5, t6, // temporary registers
s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, //
a0, a1, a2, a3, a4, a5, a6, a7 //argument registers
```

example code

```
loop:       add s6 s5 t0     --loc + i
            ld s6 s6 #0      --mem[loc + i]
            add s7 s5 t0
            addi s7 s7 #1    --loc + i + 1
            ld s7 s7 #0      --mem[loc + i + 1]
            cmp s8 s7 s6     -- s6 < s7 ?
            addi s8 s8 #1    --
            add s0 s5 t0     -- loc + i
            addi s1 s0 #1    -- loc + i + 1
            brlz s8 swap     -- labels
```

Planned experiments

- Non-pipelined versus pipelined speedup
  - Hypothesis: without dependencies IPC will be proportional to pipeline depth/number of execution units
- In-order versus OoO speedup
  - Hypothesis: notable reduction in number of cycles from better utilisation of execution units
- Branch predictor accuracy for performance for programs with repetition
  - Hypothesis: more sophisticated/higher number of states for predictor will improve accuracy of predictions. Furthermore this gives notable reduction in cycles for the same programs
- Stall reduction with register renaming
  - Hypothesis: notable reduction in false dependencies with register renaming compared to without
- Average execution unit utilisation
  - Hypothesis: higher superscalar issue width proportionally increases execution utilisation until plateau when execution units are saturated