Designing a 32-bit Pipelined CPU

Introduction

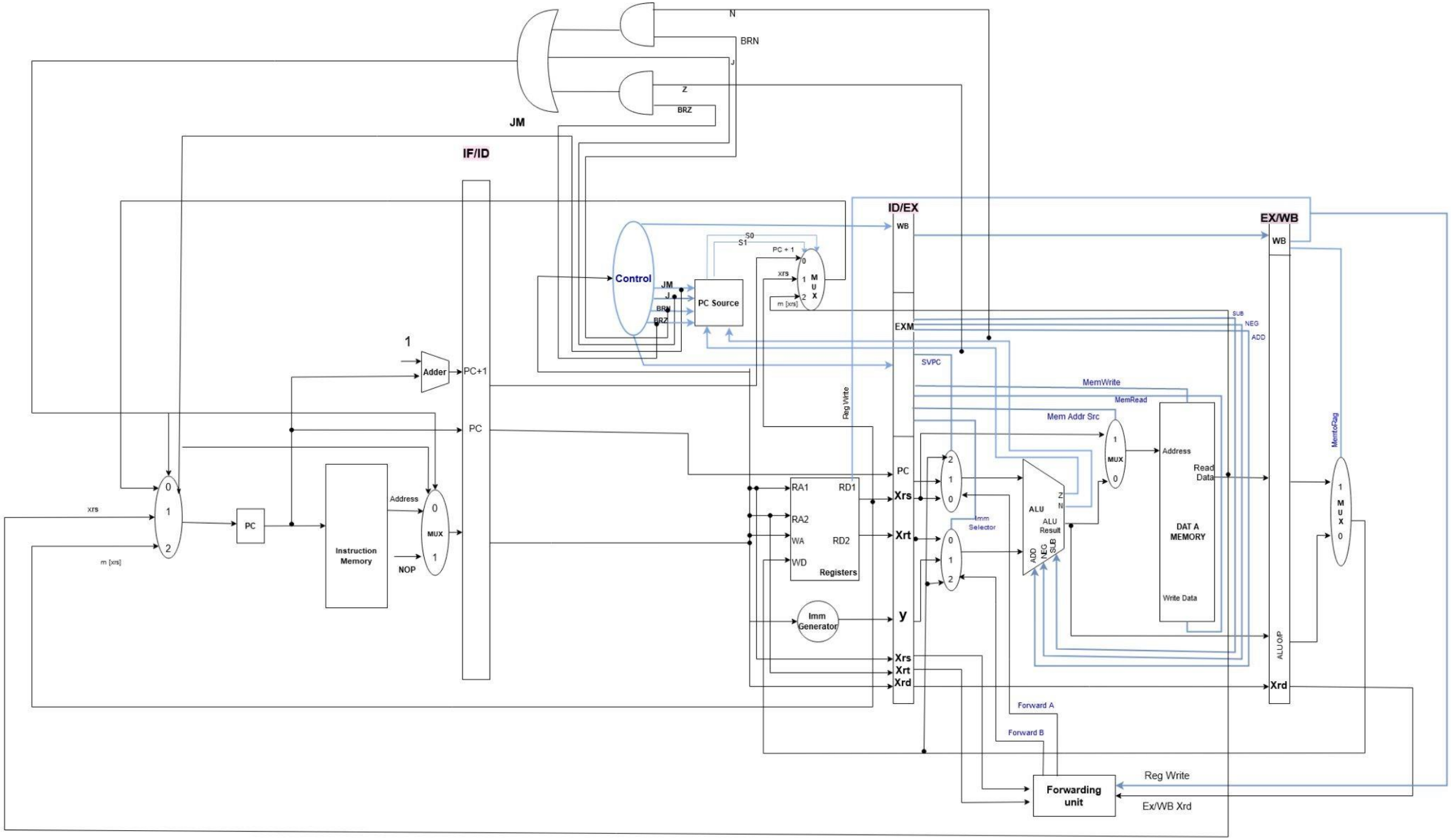
The 13 instructions are defined below.

Instruction	Symbol	Opcode	rd	rs	rt	Function
No operation	NOP	0000	×	×	×	No operation
Save PC	SVPC rd, y	1111	rd	y		xrd ← PC+y
Load	LD rd, rs	1110	rd	rs	×	xrd ← M[xrs]
Store	ST rt, rs	0011	×	rs	rt	M[xrs] ← xrt
Add	ADD rd, rs, rt	0100	rd	rs	rt	xrd ← x rs + x rt
Increment	INC rd, rs, y	0101	rd	rs	y	x rd ← x rs + y
Negate	NEG rd, rs	0110	rd	rs	×	x rd ← - x rs
Subtract	SUB rd, rs, rt	0111	rd	rs	rt	x rd ← x rs – x rt
Jump	J rs	1000	×	rs	×	PC ← x rs
Branch if zero	BRZ rs	1001	×	rs	×	PC ← x rs, if Z = 1
Jump memory	JM rs	1010	×	rs	×	PC ← M[x rs]
Branch if negative	BRN rs	1011	×	rs	×	PC ← x rs, if N = 1
MAX	MAX, rd, rs, rt	0001	rd	rs	rt	See *

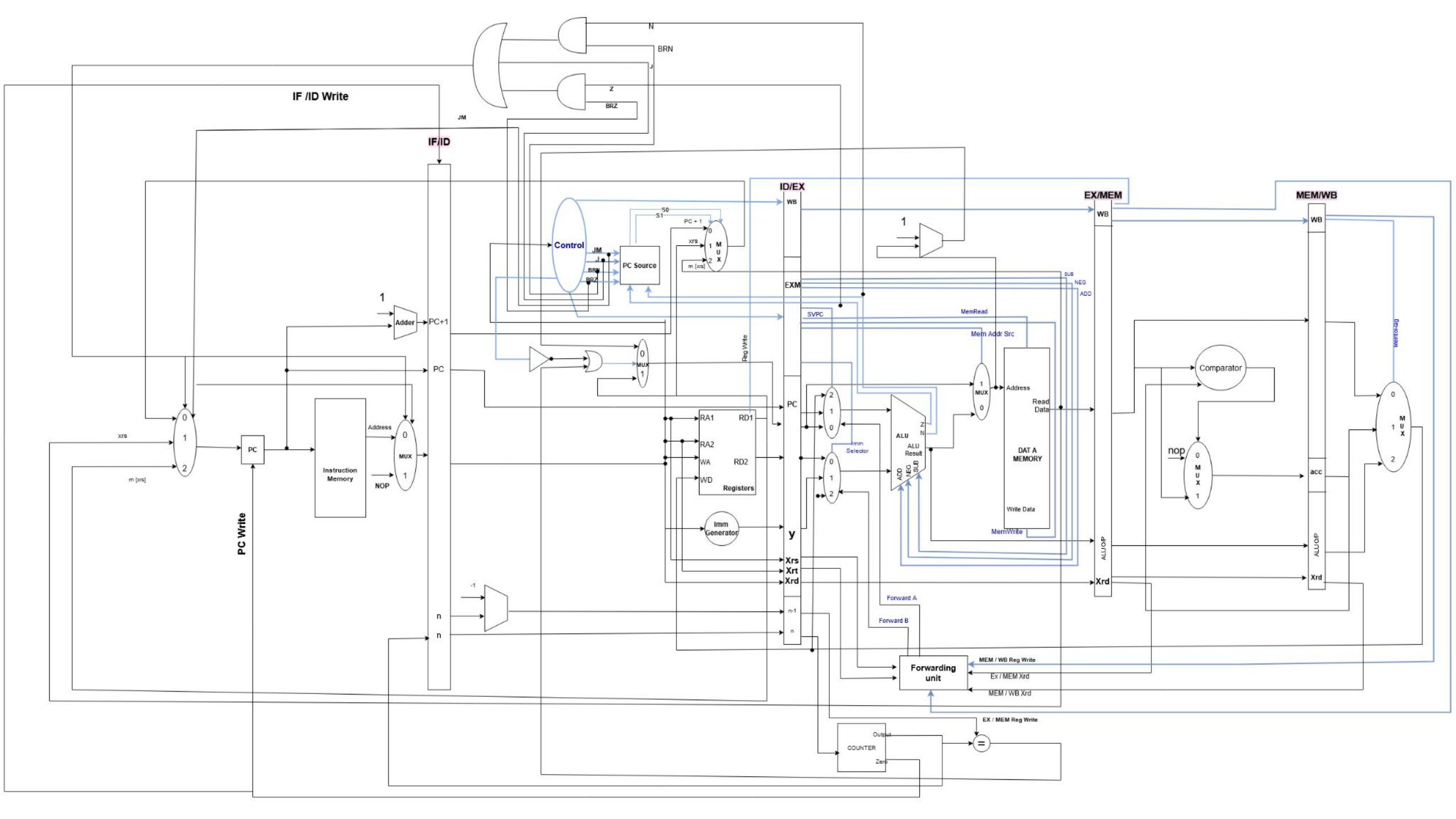
*x rd $= \max\{memory[x\ rs],\ memory[x\ rs+1],\ \ldots,\ memory[x\ rs+x\ rt-1]\}$

×: don't care

Datapath of first 12 instructions of SCU ISA with flushing and forwarding


N
BRN
J
Z
BRZ
JM
IF/ID
ID/EX
EX/WB
WB
Control
JM
J
BRN
BRZ
S0
S1
PC + 1
xrs
m [xrs]
MUX
PC Source
EXM
SVPC
Reg Write
MemWrite
MemRead
Mem Addr Src
SUB
NEG
ADD
1
Adder
PC+1
PC
xrs
m [xrs]
PC
Instruction Memory
Address
NOP
RA1
RA2
WA
WD
RD1
RD2
Registers
Imm Generator
Imm Selector
PC
Xrs
Xrt
y
Xrs
Xrt
Xrd
ALU
ALU Result
Z
N
ADD
NEG
SUB
Address
Read Data
DAT A MEMORY
Write Data
MemtoReg
ALUOP
Xrd
Forward A
Forward B
Forwarding unit
Reg Write
Ex/WB Xrd

Complete datapath for all 13 instructions of SCU ISA with flushing and forwarding


N
BRN
J
Z
BRZ
JM
IF /ID Write
IF/ID
ID/EX
EX/MEM
MEM/WB
Control
PC Source
S0
S1
PC + 1
xrs
m [xrs]
MUX
Adder
PC+1
PC
PC Write
xrs
m [xrs]
Instruction Memory
Address
NOP
RA1
RA2
WA
WD
RD1
RD2
Registers
Imm Generator
Reg Write
WB
EXM
SVPC
MemRead
Mem Addr Src
Imm Selector
ALU
ALU Result
Z
N
ADD
NEG
SUB
Address
Read Data
DAT A MEMORY
Write Data
MemWrite
Comparator
nop
acc
ALUOP
Xrs
Xrt
Xrd
n
n-1
Forward A
Forward B
Forwarding unit
MEM / WB Reg Write
Ex / MEM Xrd
MEM / WB Xrd
EX / MEM Reg Write
COUNTER
Output
Zero
=

Assembly Code: Software Loop & Modified Software Loop

```
I1:   INC x4, x0, 0
I2:   INC x3,x3,-1
I3:   ADD x5, x2, x3
I4:   SVPC X9,1
I5:   LD  X6,X2
I6:   SVPC X10,4
I7:   SUB x7,x6,x4
I8:   BRN x10
I9:   ADD x4, x6, 0
I10: INC x2,x2,1
I11:  SUB x8, x2, x5
I12:  BRN x9
I13:  NOP
```

```
I1:   INC x4, x0, 0
I2:   INC x3,x3,-1
I3:   ADD x5, x2, x3
I4:   SVPC X9,1
I5:   LD X6,X2
I6:   SUB x7,x6,x4
I7:   BRN x10
I8:   ADD x4, x6, 0
I9:   INC x2,x2,1
I10:  SVPC X10,-1
I11:  SUB x8, x2, x5
I12:  BRN x9
I13:  NOP
```

Assembly Code: Hardware Loop

```
I1:  SUB x4,x4,x4
I2:  SVPC X9, 4
I3:  ADD x3, x3, x0
I4:  BRZ x9
I5:  MAX x4,x2,x3
I6:  NOP
```

Assembly code for Loop Unrolling

```
INC x4, x0, 0
INC x3, x3, -1
ADD x5, x2, x3
SVPC x9, 1

LD x6, x2
INC x2, x2, 1
LD x11, x2
INC x2, x2, 1
LD x12, x2

SVPC x10, 4
SUB x7, x6, x4
BRN x10
ADD x4, x6, X0
```

Loop 1

```
SVPC x13, 4
SUB x7, x11, x4
BRN x13
ADD x4, x11, x0
```

Loop 2

```
SVPC x14, 4
SUB x7, x12, x4
BRN x14
ADD x4, x12, x0
```

Loop 3

```
SUB x8, x2, x5
BRN x9
NOP
```

INDIVIDUAL CPIs

BASIC PIPELINE

Instruction	R (ADD , SUB , INC, SVPC, NEG)	LD	ST	
CPI/ no dependence	1	1	1	
CPI/ with dependence	2	2	N/A	
Instruction	**BRZ**	**BRN**	**J (Jump)**	**JM (Jump Memory)**
CPI/ branch not taken	1	1	N/A	N/A
CPI/ branch taken	2	2	2	3

ADVANCED PIPELINE (Forwarding & Flushing)

Instruction	R (ADD , SUB , INC, SVPC, NEG)	LD	ST	
CPI/ no dependence	1	1	1	
CPI/ with dependence	1	1	N/A	
Instruction	**BRZ**	**BRN**	**J (Jump)**	**JM (Jump Memory)**
CPI/ branch not taken	1	1	N/A	N/A
CPI/ branch taken	2	2	2	3

INDIVIDUAL CPIs

ADVANCED PIPELINE
(FORWARING, FLUSHING, MAX INSTRUCTION)

Instruction	R (ADD , SUB , INC, SVPC, NEG)	LD	ST	
CPI/ no dependence	1	1	1	
CPI/ with dependence	2	1	N/A	
Instruction	**BRZ**	**BRN**	**J (Jump)**	**JM (Jump Memory)**
CPI/ branch not taken	1	1	N/A	N/A
CPI/ branch taken	2	2	2	3

Performance Analysis

PERFORMANCE ANALYSIS (SOFTWARE LOOP - BASIC & MODIFIED)

- **AT N=0**
  a. **CYCLE TIME = 2 ns**
  b. **INSTRUCTION COUNT = 3**
  c. **NUMBER OF CYCLES = 3(I1-I3)**
  d. **CPI = 3/3 = 1**
  e. **EXECUTION TIME = 2 * (3) = 6 ns**
- **AT N>=1**
  a. **CYCLE TIME = 2 ns**
  b. **INSTRUCTION COUNT = 8N + 4**
  c. **NUMBER OF CYCLES = 3(I1-I3) + 1(I4) + 7N(I5-I11) + 2*(N-1)(I12) + 1(I12) = 9N+3**
  d. **CPI = (9N+3)/(8N+4)**
  e. **EXECUTION TIME = 2 * (9N+3) ns**

PERFORMANCE ANALYSIS - (HARDWARE)

- At N=0
    a. CYCLE TIME = 2 ns
    b. INSTRUCTION COUNT = 5
    c. NUMBER OF CYCLES = 3(I1-I3) + 2(I4) = 5
    d. CPI = 5/5 = 1
    e. EXECUTION TIME = 2 * 5 = 10 ns
- At N >=1
    a. CYCLE TIME = 2 ns
    b. INSTRUCTION COUNT = 5
    c. NUMBER OF CYCLES = 3(I1-I3) + 1(I4) + N(I5) = N+4
    d. CPI = (N+4)/5 = 1
    e. EXECUTION TIME = 2 * (N+4) ns

Conclusion

- Comparing the Performance of the software and hardware loop we find that our speedup is almost 4 to 9 times for the the hardware instruction.

- So, hardware implementation is much more efficient than the software implementation.