CS 3220: Conte's RISC Architecture with Predication
($CR\bar{A}P$)
Document version 1.0

I. Overview

Features:

- 32-bit instructions, addresses and registers
- Predicated: 16 predicate registers
- 16 general purpose registers
- Load/Store architecture with transfer sizes byte and (32-bit) word
- No unaligned accesses permitted

II. Encoding

	IR[31:28]	IR[27:24]	IR[23:20]	IR[19:16]	IR[15:4]	IR[3:0]
Format1	Opcode	RD	RA	RB	Imm12	PR
Format2	Opcode	RD	RA	Imm16		PR

Register set:

- R0 through R15, 32-bits each
- R15 is the link register
- R0 holds a constant (unwritable) 0

Predicate register set:

- P0 through P15
- P0 holds a constant (unwritable) 0
- P1 holds a constant (unwritable) 1

III. Instruction Semantics

All instructions are executed ONLY IF Pred[PR] == 1

1. ADDI Fmt1 — Reg[RD] = Reg[RA] + signextend(Imm12)
2. ADD Fmt1 — Reg[RD] = Reg[RA] + Reg[RB]
3. ANDI Fmt1 — Reg[RD] = Reg[RA] & signextend(Imm12)
4. AND Fmt1 — Reg[RD] = Reg[RA] & Reg[RB]
5. NOT Fmt1 — Reg[RD] = ~Reg[RA]
6. PNEG Fmt1 — Pred[RD] = ~Pred[RA]
7. LSLI Fmt1 — Reg[RD] = Reg[RA] << signextend(Imm12)
8. LSL Fmt1 — Reg[RD] = Reg[RA] << Reg[RB]
9. ASRI Fmt1 — Reg[RD] = signextend(Reg[RA] >> signextend(Imm12))
10. ASR Fmt1 — Reg[RD] = signextend(Reg[RA] >> Reg[RB])
11. CMPEQI Fmt1 — Pred[RD] = (Reg[RA] == (signextend(Imm12))
12. CMPEQ Fmt1 — Pred[RD] = (Reg[RA] == Reg[RB])
13. CMPLEI Fmt1 — Pred[RD] = (Reg[RA] <= signextend(Imm12))

14. CMPLE Fmt1 Pred[RD] = (Reg[RA] <= Reg[RB])
15. LEA Fmt2 Reg[RD] = Reg[RA]+signextend(Imm16)
16. LW Fmt2 Reg[RD] = Mem[(Reg[RA] + signextend(Imm16)) & 0xFFFFFC]*
17. LB Fmt2 Reg[RD][7:0] = Mem[Reg[RA] + signextend(Imm16)]; Reg[RD][31:8] = 16b'0
18. SW Fmt2 Mem[(Reg[RA] + signextend(Imm16)) & 0xFFFFFC]* = Reg[RD]
19. SB Fmt2 Mem[Reg[RA] + signextend(Imm16)] = Reg[RD][7:0]
20. BRL Fmt2 Reg[RD] = PC; PC = PC + signextend(Imm16)
21. BR Fmt2 PC = PC + signextend(Imm16)
22. JMP Fmt2 PC = Reg[RD]
23. TRAP Fmt2 R15 = PC; PC = Memory[Imm16[7:0]]

Use RB as an opcode extension for Fmt1: Immediate form if RB == 4'b0000 (i.e., R0)
Also an opcode extension to decode between BRL and BR if BD == 4'b0000

(* to avoid unaligned access issues, word access (LW/SW) instructions force word alignment by setting bits 1:0 of the effective address to 2'b00)

All operations execute if predicate register contains a 1, else they do not execute

IV. Opcode assignments

Opcode	Bit3	Bit2	Bit1	Bit0
ADD	0	0	0	0
AND	0	0	0	1
NOT	0	0	1	0
PNEG	0	0	1	1
LSL	0	1	0	0
ASR	0	1	0	1
CMPEQ	0	1	1	0
CMPLE	0	1	1	1
LW	1	0	0	0
LB	1	0	0	1
SW	1	0	1	0
SB	1	0	1	1
BR	1	1	0	0
JMP	1	1	0	1
LEA	1	1	1	0
TRAP	1	1	1	1

	Opcode[1:0]			
Opcode[3:2]	00	01	10	11
00	ADD/ADDI	AND/ANDI	NOT	PNEG
01	LSL/LSLI	ASR/ASRI	CMPEQ/I	CMPLE/I
10	LW	LB	SW	SB
11	BR/BRL	JMP	LEA	TRAP

V. **Example program**

Problem: Copy 10 words from address 0x1000 to 0x2000

If not shown, an instruction has an implicit "?P1" (i.e., it's predicated on P1)

```
        ADDI  R1, R0, 10          // Put 10 in R1
        LEA   R2, 0x1000(R0)      // Put 0x1000 in R2
        LEA   R3, 0x2000(R0)      // Put 0x2000 in R3
Loop:   LW    R4, 0(R2)           // Get a word from the source location
        SW    R4, 0(R3)           // Store it to the destination location
        ADDI  R2, R2, #4          // Increment to next word for source
        ADDI  R3, R3, #4          // Increment to next word for dest
        ADDI  R1, R1, #-1         // Decrement R1
        CMPEQI      P2, R1, #0    // Compare R1 to 0
        PNEG P2, P2               // Invert the predicate
        BR    Loop    ?P2         // Loop if not 0
```