UV RISC Instruction Set Architecture

# Overview

* 16-bit unsigned/signed integers
* 16-bit word-addressable memory
* Simple indirect addressing (register + constant)
* 8 16-bit registers R0 through R7
  + R0 is always 0, write to R0 is ignored
  + R1 through R7 general purpose

# Instruction Descriptions

|  |  |  |
| --- | --- | --- |
| add A,B,C | Add contents of B and C, store in A | A <= B + C |
| addi A,B,imm6 | Add contents of B and immediate, store in A | A <= B + z\_ext(imm6) |
| sub A,B,C | Subtract contents of B and C, store in A | A <= B - C |
| subi A,B,imm6 | Subtract contents of B and immediate, store in A | A <= B - z\_ext(imm6) |
| mul A,B,C | Multiply unsigned contents of B and C, store low 16-bits in A, high 16-bits in R1 | A <= B \* C |
| muli A,B,imm6 | Multiply unsigned contents of B and unsigned immediate, store low 16-bits in A, high 16-bits in R1 | A <= B \* z\_ext(imm6) |
| muls A,B,C | Multiply signed contents of B and C, store low 16-bits in A, high 16-bits in R1 | A <= B \* C |
| mulsi A,B,imm6 | Multiply signed contents of B and immediate, store low 16-bits in A, high 16-bits in R1 | A <= B \* s\_ext(imm6) |
| nand A,B,C | Perform bitwise NAND of A and B, store in A | A <= ~(B & C) |
| lui A,imm10 | Store immediate in upper bits of A and clear lower bits | A <= {imm10, 000000} |
| ld A,B,imm6 | Load from memory address B + immediate, store in A | A <= M[B + z\_ext(imm6)] |
| st A,B,imm6 | Store A in memory at address B + immediate | M[B + z\_ext(imm6)] <= A |
| ba imm10 | Branch always to PC + immediate | PC <= PC + s\_ext(imm10) |
| be imm10 | Branch if equal to PC + immediate | PC <= PC + s\_ext(imm10); if Z==1 |
| blu imm10 | Branch if less than (unsigned) to PC + immediate | PC <= PC + s\_ext(imm10); if C==1 |
| bls imm10 | Branch if less than (signed) to PC + immediate | PC <= PC + s\_ext(imm10); if N!=V |

* z\_ext means zero extend (pad to 16 bits on left with zero)
* s\_ext means sign extend (pad to 16 bits on left with sign bit)

# Instruction Formats

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 3 bits | 3 bits | 3 bits | 4 bits | 3 bits |  |
| RRR1 | ALU1 | Areg | Breg | 0000 | Creg | A <= B ALU1 C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 3 bits | 3 bits | 3 bits | 4 bits | 3 bits |  |
| RRR2 | ALU2 | Areg | Breg | 0001 | Creg | A <= B ALU2 C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 3 bits | 3 bits | 3 bits | 1 bit | 6 bits |  |
| RRI | ALU1 | Areg | Breg | 1 | imm | A <= B ALU1 imm |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 3 bits | 3 bits | 3 bits | 1 bit | 6 bits |  |
| LD/ST | MEM | Areg | Breg | 1 | imm | LD: A <= M[B+imm] |
|  |  |  |  |  |  | ST: M[B+imm] <= A |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 3 bits | 3 bits | 10 bits |  |
| LUI | 110 | Areg | imm | A <= {imm, 000000} |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 3 bits | 3 bits | 10 bits |  |
| B | 111 | COND | imm | PC <= PC + imm IF COND |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ALU1 | 000 | add/addi |  | ALU2 | 000 | nand |
|  | 001 | sub/subi |  |  |  |  |
|  | 010 | mul/muli |  |  |  |  |
|  | 011 | muls/mulsi |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| COND | 000 | ba |  | MEM | 100 | ld |
|  | 001 | be |  |  | 101 | st |
|  | 010 | blu |  |  |  |  |
|  | 011 | bls |  |  |  |  |

# Example Program

Divide 24 by 7 using repeated subtraction:

|  |  |  |  |
| --- | --- | --- | --- |
| Address | Binary instruction | Assembly instruction | |
| 0 | 16'b110\_010\_0000000000 | lhi | R2,0 |
| 1 | 16'b000\_010\_010\_1\_011000 | add | R2,R2,24 // R2=24 |
| 2 | 16'b110\_011\_0000000000 | lhi | R3,0 |
| 3 | 16'b000\_011\_011\_1\_000111 | add | R3,R3,7 // R3=7 |
| 4 | 16'b000\_111\_000\_1\_000000 | add | R7,R0,0 // R7=0 |
| 5 | 16'b001\_010\_010\_0000\_011 | sub | R2,R2,R3 |
| 6 | 16'b111\_011\_0000000011 | bls | +3 (=9) |
| 7 | 16'b000\_111\_111\_1\_000001 | add | R7,R7,1 |
| 8 | 16'b111\_000\_1111111101 | ba | -3 (=5) |
| 9 | 16'b111\_000\_0000000000 | ba | +0 (=9) |

# UV RISC Datapath

