

TSK 1113 Computer Organisation and Architecture

ARC Instruction

"nop"
"halt"
"sethi"
"be" "bcs" "bneg" "bvs" "ba" "bne" "bcc" "bpos" "bvc"
"call"
"jmp1"
"addcc" "andcc" "subcc" "orcc" "orncc" "xorcc"
"srl" "sll" "sra"
"add" "sub" "and" "or" "orn" "xor"
"ld" "st"
".dwb"
".begin"
".end"
".org"
".equ"

"nop" performs no instruction but increments the program counter.

"halt" stops the simulator.

"sethi" sets the high 22 bits and zeros the low 10 bits of a register. If the operand is 0 and the register is %r0, then the instruction behaves as a no-op (nop).

Example usage: sethi 0x304F15, %r1. Meaning: Set the high 22 bits of %r1 to 0x304F15 and set the lower 10 bits to zero.

"be" branch on equal to zero. If the z condition code is 1, then branch to the address represented by the label which is the instruction operand.

Example usage: be label. Meaning: Branch to label if Z is 1.

"bcs" branch on C set. If the c condition code is 1, then branch to the address represented by the label which is the instruction operand.

Example usage: bcs label. Meaning: Branch to label if C is 1.

"bcc" branch on carry clear. If the c condition code is 0, then branch to the address represented by the label which is the instruction operand.

Example usage: bcc label. Meaning: Branch to label if C is 0.

"bneg" branch on negative. If the n condition code is 1, then branch to the address represented by the label which is the instruction operand.

Example usage: bneg label. Meaning: Branch to label if N is 1.

"bvs" branch on signed overflow. If the v condition code is 1, then branch to the address represented by the label which is the instruction operand.

Example usage: bvs label. Meaning: Branch to label if V is 1.

"bvc" branch on no signed overflow. If the v condition code is 0, then branch to the address represented by the label which is the instruction operand.

Example usage: bvc label. Meaning: Branch to label if V is 0.

"bne" branch on not equal. Branch if not equal to zero to the address represented by the label which is the instruction operand.

Example usage: `bne label`. Meaning: Branch to label if not equal to zero.

"bpos" branch on positive. If the condition codes signal a positive result, branch to the address represented by the label which is the instruction operand.

Example usage: `bpos label`. Meaning: Branch if positive.

"ba" branch always. Always branch to the address represented by the label which is the instruction operand.

Example usage: `ba label`. Meaning: Always branch to label.

"call" call a subroutine and store the address of the current instruction in `%r15`. The instruction operand is the address of the subroutine and is stored as a 30 bit displacement in the call instruction format.

Example usage: `call sub_r`. Meaning: Call the subroutine located at `sub_r`.

"addcc" and **"add"** adds the source operands into the destination operand using two's complement arithmetic.

"addcc" sets the condition codes according to the result.

Example usage: `add %r1, %r2, %r4`. Meaning: `%r4 = %r1 + %r2`

Example usage: `addcc %r1, 2, %r2`. Meaning: `%r2 = %r2 + 2`

"subcc" and **"sub"** perform integer subtraction on the source operands and put result into the destination operand using two's complement arithmetic. **"subcc"** sets the condition codes according to the result.

Example usage: `sub %r1, %r2, %r4`. Meaning: `%r4 = %r1 - %r2`

Example usage: `subcc %r1, 2, %r2`. Meaning: `%r2 = %r2 - 2`

"andcc" and **"and"** bitwise AND the source operands into the destination operand. **"andcc"** sets the N and Z condition codes according to the result.

Example usage: `and %r1, %r2, %r4`. Meaning: `%r4 = %r1 AND %r2`

"orcc" and **"or"** bitwise OR the source operands into the destination operand. **"orcc"** sets the N and Z condition codes according to the result.

Example usage: `or %r1, %r2, %r4`. Meaning: `%r4 = %r1 OR %r2`

Example usage: `or %r1, 1, %r2`. Meaning: `%r2 = %r1 OR 1`

"orncc" and **"orn"** bitwise NOR the source operands into the destination operand. **"orncc"** will set the N and Z condition codes according to the result.

Example usage: `orncc %r1, %r0, %r1`. Meaning: Complement `%r1`.

"xor" and **"xorcc"** bitwise XOR (exclusive OR) the source operands into the destination operand. **"xorcc"** will set the N and Z condition codes according to the result.

Example usage: `xorcc %r1, %r0, %r1`. Meaning: `%r1 = %r1 XOR %r0`.

"srl" shifts a register to the right by 0-31 bits. The vacant bit positions in the left side of the shifted register are filled with 0's.

Example usage: `srl %r1, 3, %r4`. Meaning: Shift `%r1` right by 3 bits and store in `%r4`.

Example usage: `srl %r1, %r4, %r5`. Meaning: Shift `%r1` right by the value stored in `%r4` and store in `%r5`.

"sll" shifts a register to the left by 0-31 bits. The vacant bit positions in the right side of the shifted register are filled with 0's.

Example usage: `sll %r1, 3, %r4`. Meaning: Shift `%r1` left by 3 bits and store in `%r4`.

Example usage: `sll %r1, %r4, %r5`. Meaning: Shift `%r1` left by the value stored in `%r4` and store in `%r5`.

"sra" shifts a register to the right by 0-31 bits. The sign bit is replicated as the value is shifted right.

Example usage: `sra %r1, 3, %r4`. Meaning: Shift `%r1` right by 3 bits and store in `%r4`.

Example usage: `sra %r1, %r4, %r5`. Meaning: Shift `%r1` right by the value stored in `%r4` and store in `%r5`.

"ld" load a register from main memory. The memory address must be aligned on a word boundary.

Example usages: `ld [x], %r1`

```
ld [x], %r0, %r1
ld %r0+x, %r1
```

Meaning: Copy the contents of memory location x into register %r1. (%r0 is always exactly zero.)

"st" store a register into main memory. The memory address must be aligned on a word boundary.

Example usages: st %r1, [x]
 st %r1, %r0 + x
 st %r1, %r0, x

Meaning: Store the contents of %r1 into the memory location x. (%r0 is always exactly zero.)

"jmpl" unconditional, register indirect control transfer. Jump to a new address and store the address of the current instruction in the destination register.

Example usage: jmpl %r15 + 4, %r2

Meaning: Set the program counter to the contents of %r15 + 4. The current address is stored in to %r2.

The following pseudo-operations are supported:

.dwb n Define Word Block: reserve storage for a block of n words.

.begin, .end start and stop assembly, respectively.

Symbol **.equ** value Equate value to Symbol in the symbol table.

NOTE: Symbol should ***NOT*** be followed by a colon(:) because it is not a program label.

Example usage:

WS .equ 32

.org Symbol Change location counter to the address specified by Symbol.

Example usage:

 .org 0x1000 ! next instruction will be assembled at location 0x1000.

Start .equ 0x2000

 .org Start ! next instruction will be assembled at location 0x2000

Comments begin with the ! symbol and continue to the end of line.

Example usage:

 nop ! This is the no operation instruction.