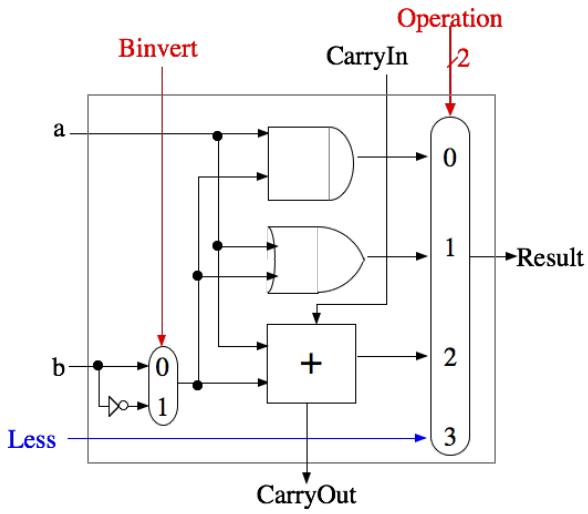
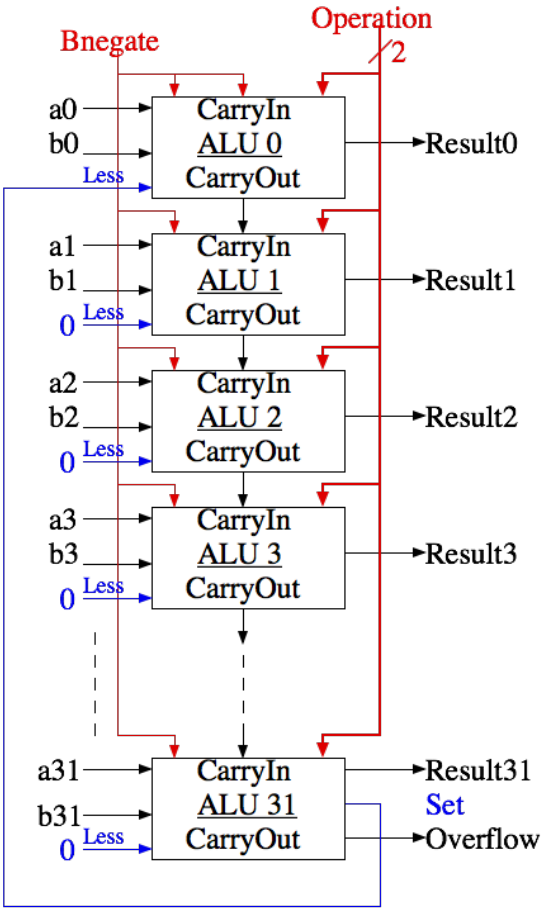
Formula Sheet

Example: one-bit ALU



Example: 32-bit ALU

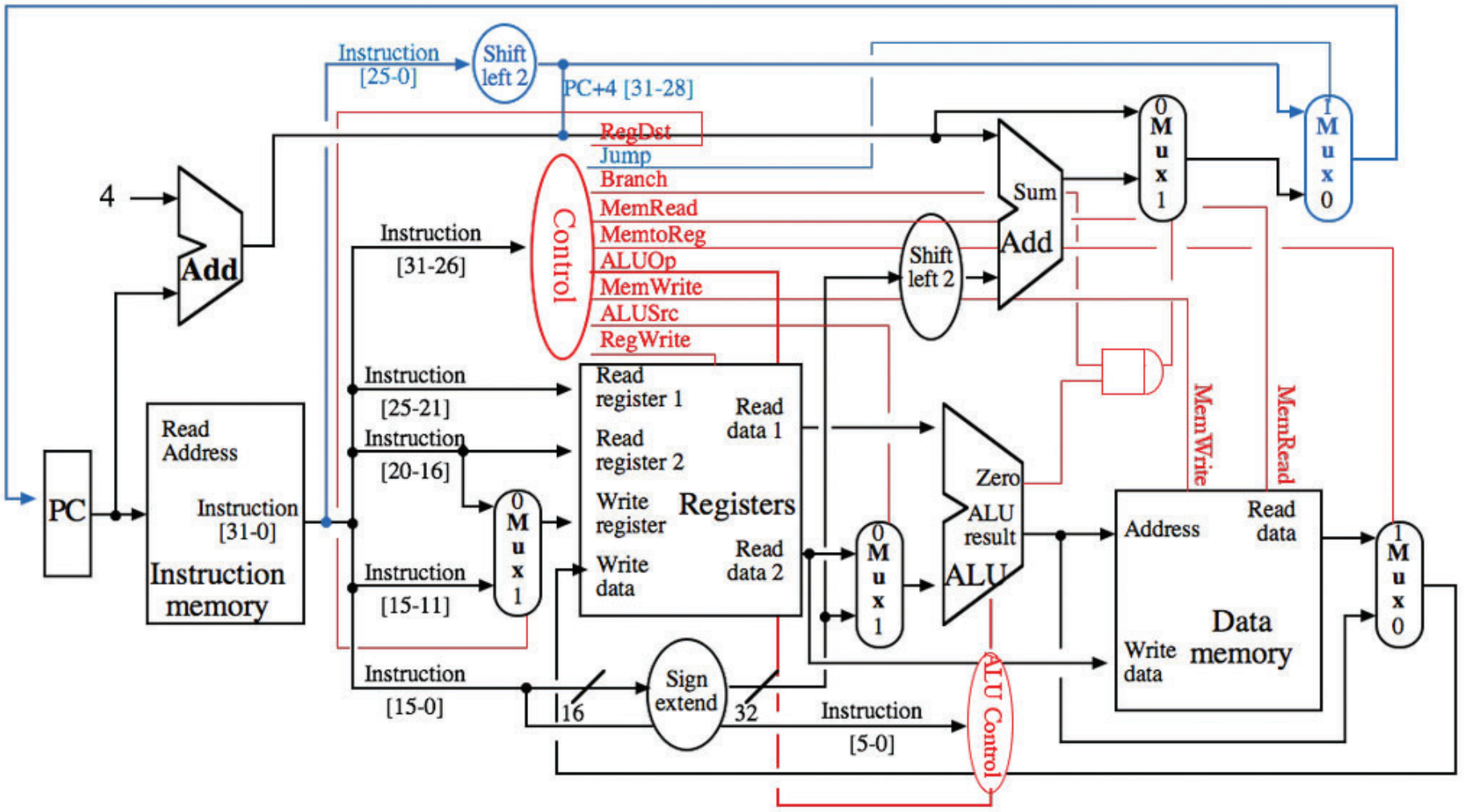


Standard Prologue

```
addiu   $sp, $sp, -24
sw      $fp, 0($sp)
sw      $ra, 4($sp)
addiu   $fp, $sp, 20
```

Standard Epilogue

```
lw      $ra, 4($sp)
lw      $fp, 0($sp)
addiu   $sp, $sp, 24
jr      $ra
```

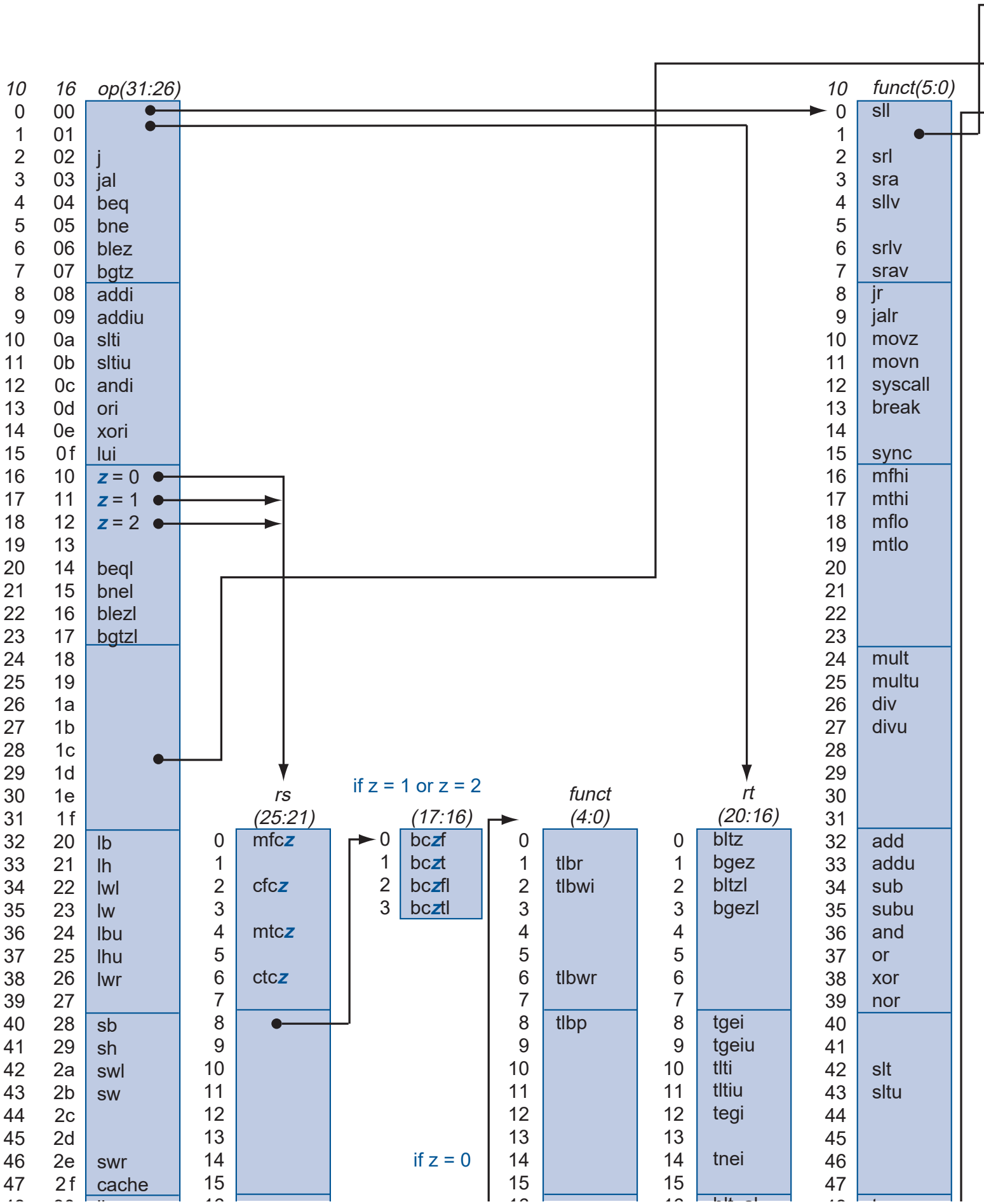

Instruction [25-0]
Shift left 2
PC+4 [31-28]
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
4
Add
Instruction [31-26]
Instruction [25-21]
Instruction [20-16]
Instruction [15-11]
Instruction [15-0]
Instruction [5-0]
PC
Read Address
Instruction [31-0]
Instruction memory
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
0 M u x 1
Shift left 2
Sum
Add
0 M u x 1
1 M u x 0
Zero
ALU
ALU result
Address
Read data
Write data
Data memory
MemWrite
MemRead
16
Sign extend
32
ALU Control

Register name	Number	Usage
$zero	0	constant 0
$at	1	reserved for assembler
$v0	2	expression evaluation and results of a function
$v1	3	expression evaluation and results of a function
$a0	4	argument 1
$a1	5	argument 2
$a2	6	argument 3
$a3	7	argument 4
$t0	8	temporary (not preserved across call)
$t1	9	temporary (not preserved across call)
$t2	10	temporary (not preserved across call)
$t3	11	temporary (not preserved across call)
$t4	12	temporary (not preserved across call)
$t5	13	temporary (not preserved across call)
$t6	14	temporary (not preserved across call)
$t7	15	temporary (not preserved across call)
$s0	16	saved temporary (preserved across call)
$s1	17	saved temporary (preserved across call)
$s2	18	saved temporary (preserved across call)
$s3	19	saved temporary (preserved across call)
$s4	20	saved temporary (preserved across call)
$s5	21	saved temporary (preserved across call)
$s6	22	saved temporary (preserved across call)
$s7	23	saved temporary (preserved across call)
$t8	24	temporary (not preserved across call)
$t9	25	temporary (not preserved across call)
$k0	26	reserved for OS kernel
$k1	27	reserved for OS kernel
$gp	28	pointer to global area
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address (used by function call)

FIGURE A.6.1 MIPS registers and usage convention.

the stack pointer. The executing procedure uses the frame pointer to qu
access values in its stack frame. For example, an argument in the stack fram
be loaded into register `$v0` with the instruction

```
lw $v0, 0($fp)
```


10 16 op(31:26)
0 00
1 01
2 02 j
3 03 jal
4 04 beq
5 05 bne
6 06 blez
7 07 bgtz
8 08 addi
9 09 addiu
10 0a slti
11 0b sltiu
12 0c andi
13 0d ori
14 0e xori
15 0f lui
16 10 z = 0
17 11 z = 1
18 12 z = 2
19 13
20 14 beql
21 15 bnel
22 16 blezl
23 17 bgtzl
24 18
25 19
26 1a
27 1b
28 1c
29 1d
30 1e
31 1f
32 20 lb
33 21 lh
34 22 lwl
35 23 lw
36 24 lbu
37 25 lhu
38 26 lwr
39 27
40 28 sb
41 29 sh
42 2a swl
43 2b sw
44 2c
45 2d
46 2e swr
47 2f cache
rs (25:21)
0 mfcz
1
2 cfcz
3
4 mtcz
5
6 ctcz
7
8
9
10
11
12
13
14
15
if z = 1 or z = 2
(17:16)
0 bczf
1 bczt
2 bczfl
3 bcztl
if z = 0
funct (4:0)
0
1 tlbr
2 tlbwi
3
4
5
6 tlbwr
7
8 tlbp
9
10
11
12
13
14
15
rt (20:16)
0 bltz
1 bgez
2 bltzl
3 bgezl
4
5
6
7
8 tgei
9 tgeiu
10 tlti
11 tltiu
12 tegi
13
14 tnei
15
10 funct(5:0)
0 sll
1
2 srl
3 sra
4 sllv
5
6 srlv
7 srav
8 jr
9 jalr
10 movz
11 movn
12 syscall
13 break
14
15 sync
16 mfhi
17 mthi
18 mflo
19 mtlo
20
21
22
23
24 mult
25 multu
26 div
27 divu
28
29
30
31
32 add
33 addu
34 sub
35 subu
36 and
37 or
38 xor
39 nor
40
41
42 slt
43 sltu
44
45
46
47