Function calls

High-Level Code

```
int main()
{
  int y;

  . . .

  y = diffofsums(2, 3, 4, 5);

  . . .
}

int diffofsums(int f, int g, int h, int i)
{
  int result;

  result = (f + g) - (h + i);
  return result;
}
```

MIPS Assembly Code

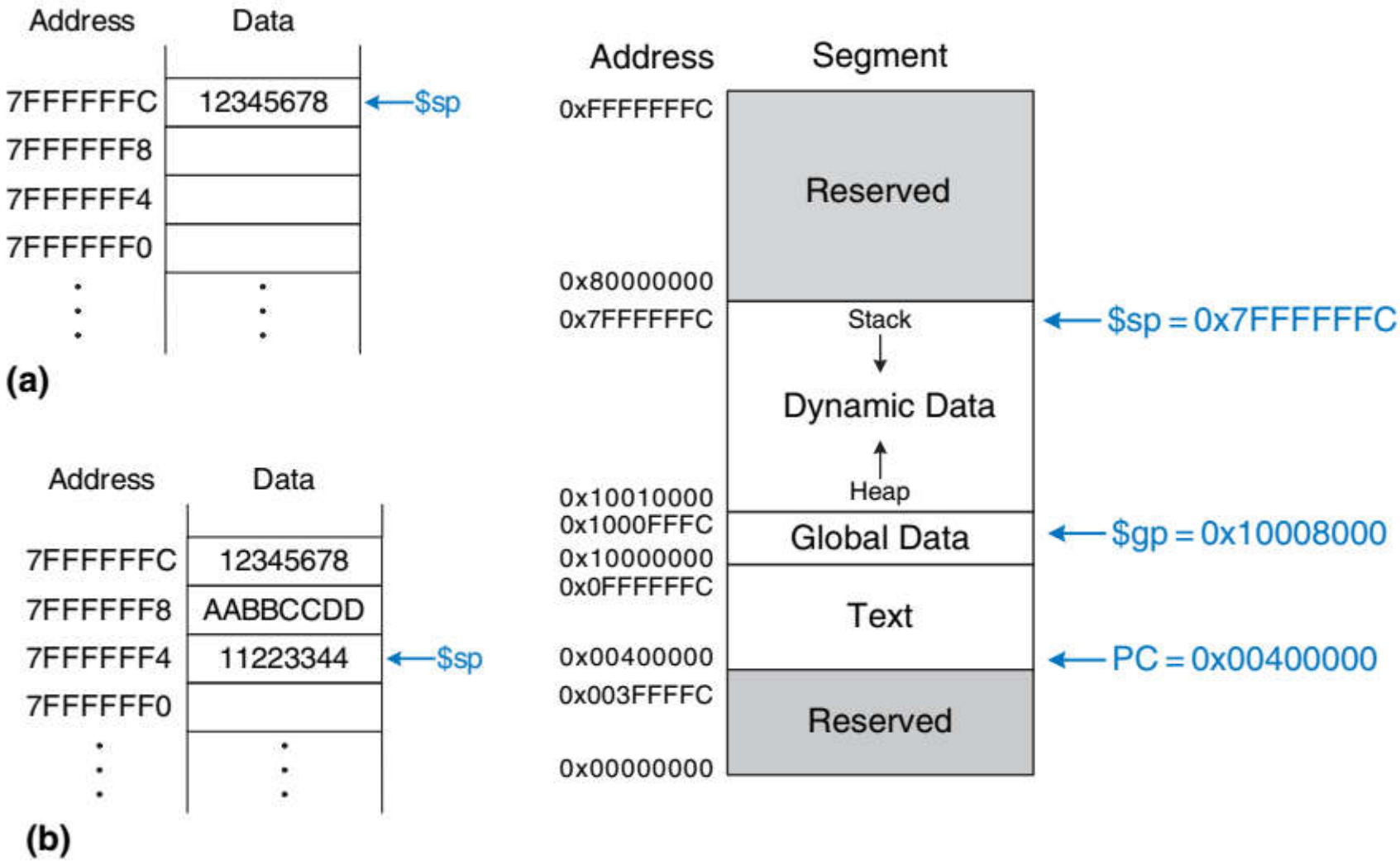
```
# $s0 = y
main:
  . . .
  addi $a0, $0, 2     # argument 0 = 2
  addi $a1, $0, 3     # argument 1 = 3
  addi $a2, $0, 4     # argument 2 = 4
  addi $a3, $0, 5     # argument 3 = 5
  jal  diffofsums     # call function
  add  $s0, $v0, $0   # y = returned value
  . . .

# $s0 = result
diffofsums:
  add $t0, $a0, $a1   # $t0 = f + g
  add $t1, $a2, $a3   # $t1 = h + i
  sub $s0, $t0, $t1   # result = (f + g) - (h + i)
  add $v0, $s0, $0    # put return value in $v0
  jr  $ra             # return to caller
```

Name	Number	Use
$0	0	the constant value 0
$at	1	assembler temporary
$v0–$v1	2–3	function return value
$a0–$a3	4–7	function arguments
$t0–$t7	8–15	temporary variables
$s0–$s7	16–23	saved variables
$t8–$t9	24–25	temporary variables
$k0–$k1	26–27	operating system (OS) temporaries
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	function return address

$a0 – $a3 – arguments
$v0 – $v1 – results

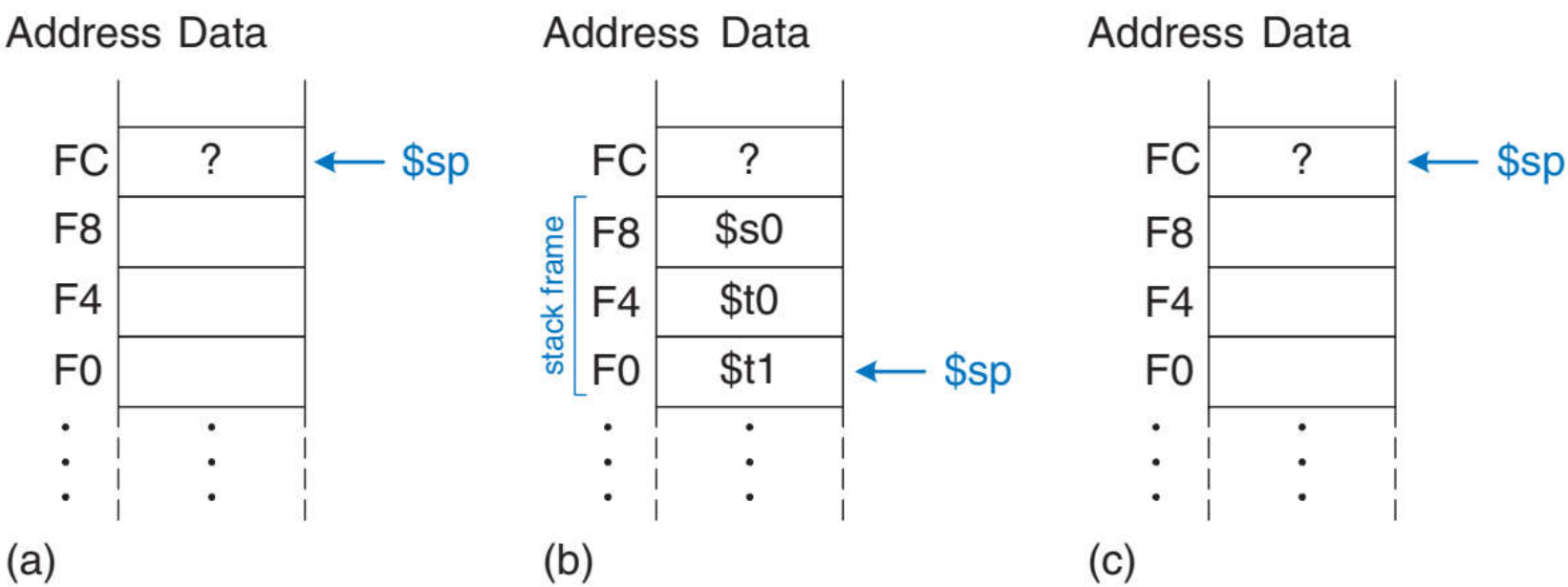
jal – jump and link
jr – jump register (address)

Memory Map and Stack

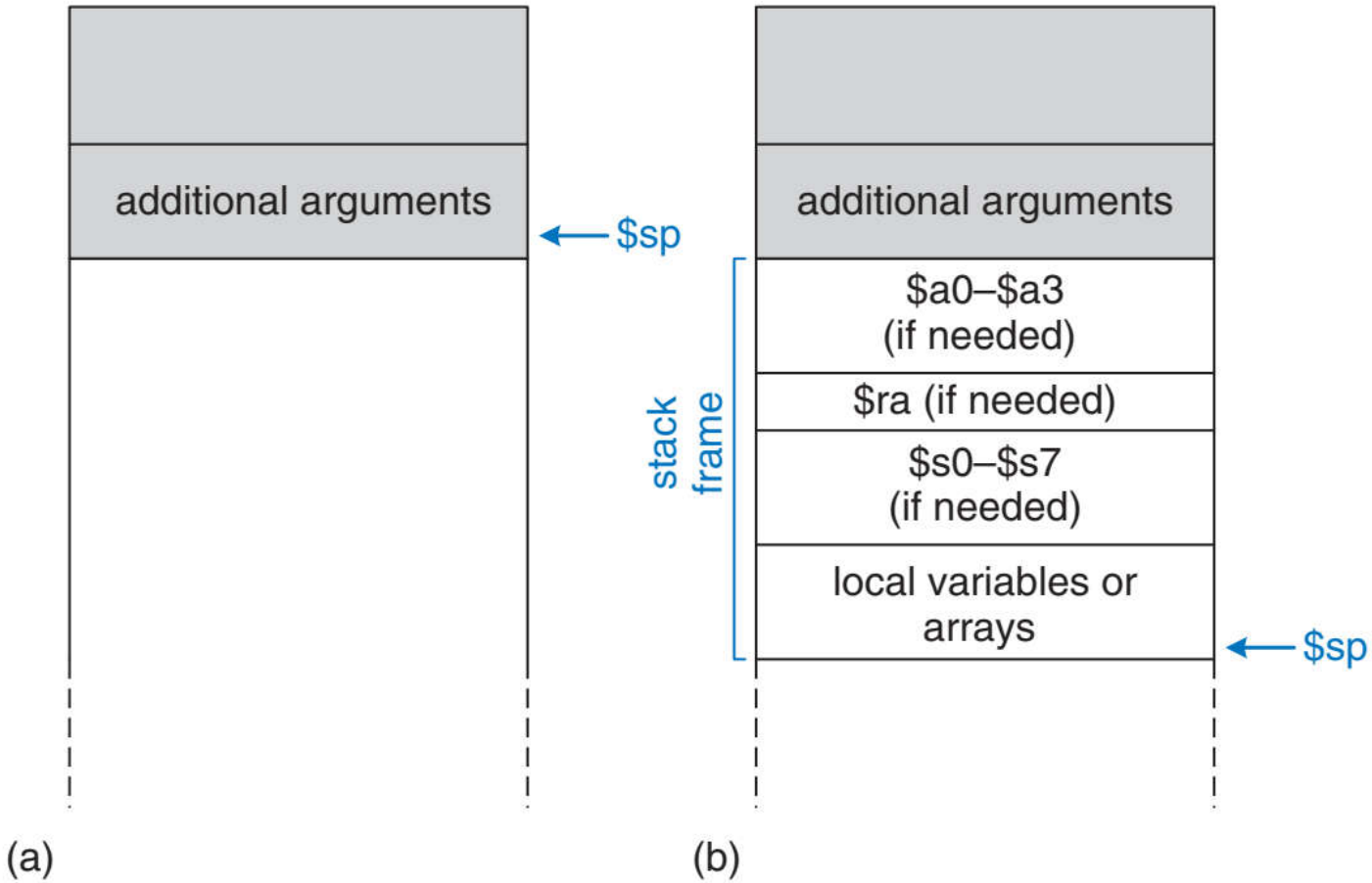


Function in assembly

```
# $s0 = result
diffofsums:
addi $sp, $sp, -12      # make space on stack to store three registers
sw $s0, 8($sp)          # save $s0 on stack
sw $t0, 4($sp)          # save $t0 on stack
sw $t1, 0($sp)                     # save $t1 on stack
add $t0, $a0, $a1       # $t0 = f + g
add $t1, $a2, $a3       # $t1 = h + i
sub $s0, $t0, $t1       # result = (f + g) - (h + i)
add $v0, $s0, $0        # put return value in $v0
lw $t1, 0($sp)                     # restore $t1 from stack
lw $t0, 4($sp)                     # restore $t0 from stack
lw $s0, 8($sp)          # restore $s0 from stack
addi $sp, $sp, 12       # deallocate stack space
jr $ra                  # return to caller
```



Stack operations



Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8   # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0        # put arguments a0 – a3
jal   subroutine    # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8    # space for ra and fp
sw   $ra,  4($sp)    # push ra
sw   $fp,  0($sp)    # push fp
addi $sp, $sp, -12   # this may vary
sw   $s0,  8($sp)    # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16   # local vars a,b,i,j
addi $sp, $fp,  0    # initialize sp
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
SP	0x7FFFFFEC	main a
	0x7FFFFFE8	
	0x7FFFFFE4	
	0x7FFFFFE0	
	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8    # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0         # put arguments a0 – a3
jal   subroutine     # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8     # space for ra and fp
sw   $ra,  4($sp)     # push ra
sw   $fp,  0($sp)     # push fp
addi $sp, $sp, -12    # this may vary
sw   $s0,  8($sp)     # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16    # local vars a,b,i,j
addi $sp, $fp,  0     # initialize sp
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	
SP	0x7FFFFFE4	
	0x7FFFFFE0	
	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8   # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0        # put arguments a0 – a3
jal   subroutine    # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8    # space for ra and fp
sw   $ra,  4($sp)    # push ra
sw   $fp,  0($sp)    # push fp
addi $sp, $sp, -12   # this may vary
sw   $s0,  8($sp)    # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16   # local vars a,b,i,j
addi $sp, $fp,  0    # initialize sp
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
SP	0x7FFFFFE4	
	0x7FFFFFE0	
	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8   # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0        # put arguments a0 – a3
jal   subroutine    # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8    # space for ra and fp
sw   $ra,  4($sp)    # push ra
sw   $fp,  0($sp)    # push fp
addi $sp, $sp, -12   # this may vary
sw   $s0,  8($sp)    # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16   # local vars a,b,i,j
addi $sp, $fp,  0    # initialize sp
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
SP	0x7FFFFFE4	t3
	0x7FFFFFE0	
	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8   # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0        # put arguments a0 – a3
jal   subroutine    # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8    # space for ra and fp
sw   $ra,  4($sp)    # push ra
sw   $fp,  0($sp)    # push fp
addi $sp, $sp, -12   # this may vary
sw   $s0,  8($sp)    # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16   # local vars a,b,i,j
addi $sp, $fp,  0    # initialize sp
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	
SP	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8    # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0         # put arguments a0 – a3
jal   subroutine     # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8     # space for ra and fp
sw   $ra,  4($sp)     # push ra
sw   $fp,  0($sp)     # push fp
addi $sp, $sp, -12    # this may vary
sw   $s0,  8($sp)     # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16    # local vars a,b,i,j
addi $sp, $fp,  0     # initialize sp
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
SP	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8    # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0         # put arguments a0 – a3
jal   subroutine     # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8     # space for ra and fp
sw   $ra,  4($sp)     # push ra
sw   $fp,  0($sp)     # push fp
addi $sp, $sp, -12    # this may vary
sw   $s0,  8($sp)     # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16    # local vars a,b,i,j
addi $sp, $fp,  0     # initialize sp
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
SP	0x7FFFFFDC	fp
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8    # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0         # put arguments a0 – a3
jal   subroutine     # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8     # space for ra and fp
sw   $ra,  4($sp)     # push ra
sw   $fp,  0($sp)     # push fp
addi $sp, $sp, -12    # this may vary
sw   $s0,  8($sp)     # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16    # local vars a,b,i,j
addi $sp, $fp,  0     # initialize sp
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	
	0x7FFFFFD4	
SP	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8    # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0         # put arguments a0 – a3
jal   subroutine     # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8     # space for ra and fp
sw   $ra,  4($sp)     # push ra
sw   $fp,  0($sp)     # push fp
addi $sp, $sp, -12    # this may vary
sw   $s0,  8($sp)     # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16    # local vars a,b,i,j
addi $sp, $fp,  0     # initialize sp
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	s0
	0x7FFFFFD4	
SP	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8   # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0        # put arguments a0 – a3
jal   subroutine    # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8    # space for ra and fp
sw   $ra,  4($sp)    # push ra
sw   $fp,  0($sp)    # push fp
addi $sp, $sp, -12   # this may vary
sw   $s0,  8($sp)    # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16   # local vars a,b,i,j
addi $sp, $fp,  0    # initialize sp
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	s0
	0x7FFFFFD4	s3
SP	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8    # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0         # put arguments a0 – a3
jal   subroutine     # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8     # space for ra and fp
sw   $ra,  4($sp)     # push ra
sw   $fp,  0($sp)     # push fp
addi $sp, $sp, -12    # this may vary
sw   $s0,  8($sp)     # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16    # local vars a,b,i,j
addi $sp, $fp,  0     # initialize sp
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	s0
	0x7FFFFFD4	s3
SP	0x7FFFFFD0	s5
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8   # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0        # put arguments a0 – a3
jal   subroutine    # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8    # space for ra and fp
sw   $ra,  4($sp)    # push ra
sw   $fp,  0($sp)    # push fp
addi $sp, $sp, -12   # this may vary
sw   $s0,  8($sp)    # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16   # local vars a,b,i,j
addi $sp, $fp,  0    # initialize sp
```

	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	s0
	0x7FFFFFD4	s3
SP	0x7FFFFFD0	s5
	0x7FFFFFCC	a
	0x7FFFFFC8	b
	0x7FFFFFC4	i
FP	0x7FFFFFC0	j

Frame-based Linkage Convention

Calling a subroutine by caller

```
# store temporary registers on the stack
addi $sp, $sp, -8   # this may vary
sw   $t0, 4($sp)
sw   $t3, 0($sp)
mov  $a0, $0        # put arguments a0 – a3
jal   subroutine    # call subroutine
```

Subroutine prolog

```
# store temporary registers on the stack
subroutine:
addi $sp, $sp, -8    # space for ra and fp
sw   $ra,  4($sp)    # push ra
sw   $fp,  0($sp)    # push fp
addi $sp, $sp, -12   # this may vary
sw   $s0,  8($sp)    # push s0, s3, s5
sw   $s3,  4($sp)
sw   $s5,  0($sp)
addi $fp, $sp, -16   # local vars a,b,i,j
addi $sp, $fp,  0    # initialize sp
```

	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	s0
	0x7FFFFFD4	s3
	0x7FFFFFD0	s5
	0x7FFFFFCC	a
	0x7FFFFFC8	b
	0x7FFFFFC4	i
SP, FP	0x7FFFFFC0	j

Frame-based Linkage Convention

Subroutine execution

```
# use any t, s, register to perform operations
# use fp to access the local variables
lw   $t0, 4($fp)     #  get i value
addi $t0, $t0, 1     #  i++
sw   $t0, 4($fp)     #  store i value
addi $sp, $sp, -4    # allocate stack
sw   $t4, 0($sp)     # store t4 value
...
lw   $t4, 0($sp)     # load t4 value
addi $sp, $sp, 4     # deallocate stack
```

	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	s0
	0x7FFFFFD4	s3
	0x7FFFFFD0	s5
	0x7FFFFFCC	a
	0x7FFFFFC8	b
	0x7FFFFFC4	i+1
SP,FP	0x7FFFFFC0	j
	0x7FFFFFBC	

Frame-based Linkage Convention

Subroutine execution

```
# use any t, s, register to perform operations
# use fp to access the local variables
lw   $t0, 4($fp)     #  get i value
addi $t0, $t0, 1     #  i++
sw   $t0, 4($fp)     #  store i value
addi $sp, $sp, -4    # allocate stack
sw   $t4, 0($sp)     # store t4 value
...
lw   $t4, 0($sp)     # load t4 value
addi $sp, $sp, 4     # deallocate stack
```

	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	s0
	0x7FFFFFD4	s3
	0x7FFFFFD0	s5
	0x7FFFFFCC	a
	0x7FFFFFC8	b
	0x7FFFFFC4	i
FP	0x7FFFFFC0	j
SP	0x7FFFFFBC	

Frame-based Linkage Convention

Subroutine execution

```
# use any t, s, register to perform operations
# use fp to access the local variables
lw   $t0, 4($fp)     #  get i value
addi $t0, $t0, 1     #  i++
sw   $t0, 4($fp)     #  store i value
addi $sp, $sp, -4    # allocate stack
sw   $t4, 0($sp)     # store t4 value
...
lw   $t4, 0($sp)     # load t4 value
addi $sp, $sp, 4     # deallocate stack
```

	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	s0
	0x7FFFFFD4	s3
	0x7FFFFFD0	s5
	0x7FFFFFCC	a
	0x7FFFFFC8	b
	0x7FFFFFC4	i
FP	0x7FFFFFC0	j
SP	0x7FFFFFBC	t4

Frame-based Linkage Convention

Subroutine execution

```
# use any t, s, register to perform operations
# use fp to access the local variables
lw   $t0, 4($fp)    #  get i value
addi $t0, $t0, 1    #  i++
sw   $t0, 4($fp)    #  store i value
addi $sp, $sp, -4   # allocate stack
sw   $t4, 0($sp)    # store t4 value
...
lw   $t4, 0($sp)    # load t4 value
addi $sp, $sp, 4    # deallocate stack
```

	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	s0
	0x7FFFFFD4	s3
	0x7FFFFFD0	s5
	0x7FFFFFCC	a
	0x7FFFFFC8	b
	0x7FFFFFC4	i
SP,FP	0x7FFFFFC0	j
	0x7FFFFFBC	

Frame-based Linkage Convention

Subroutine epilog

```
# restore temporary registers from the stack
mov  $v0,  8($fp)     # return a value
addi $sp,  $fp,  16   # deallocate local vars
lw   $s0,  8($sp)     # pop s0, s3, s5
lw   $s3,  4($sp)
lw   $s5,  0($sp)
addi $sp,  $sp, 12    # deallocate S reg space
lw   $fp,  0($sp)     # pop fp
lw   $ra,  4($sp)     # pop ra
addi $sp,  $sp, 8     # deallocate ra and fp
jr   $ra              # jump register
```

	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	s0
	0x7FFFFFD4	s3
	0x7FFFFFD0	s5
	0x7FFFFFCC	a
	0x7FFFFFC8	b
	0x7FFFFFC4	i
SP,FP	0x7FFFFFC0	j
	0x7FFFFFBC	

Frame-based Linkage Convention

Subroutine epilog

```
# restore temporary registers from the stack
mov  $v0,  8($fp)     # return a value
addi $sp,  $fp,  16   # deallocate local vars
lw   $s0,  8($sp)     # pop s0, s3, s5
lw   $s3,  4($sp)
lw   $s5,  0($sp)
addi $sp,  $sp, 12    # deallocate S reg space
lw   $fp,  0($sp)     # pop fp
lw   $ra,  4($sp)     # pop ra
addi $sp,  $sp, 8     # deallocate ra and fp
jr   $ra              # jump register
```

	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	s0
	0x7FFFFFD4	s3
SP	0x7FFFFFD0	s5
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
FP	0x7FFFFFC0	
	0x7FFFFFBC	

Frame-based Linkage Convention

Subroutine epilog

```
# restore temporary registers from the stack
mov  $v0,  8($fp)     # return a value
addi $sp,  $fp,  16   # deallocate local vars
lw   $s0,  8($sp)     # pop s0, s3, s5
lw   $s3,  4($sp)
lw   $s5,  0($sp)
addi $sp,  $sp, 12    # deallocate S reg space
lw   $fp,  0($sp)     # pop fp
lw   $ra,  4($sp)     # pop ra
addi $sp,  $sp, 8     # deallocate ra and fp
jr   $ra              # jump register
```

	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
	0x7FFFFFDC	fp
	0x7FFFFFD8	
	0x7FFFFFD4	s3
SP	0x7FFFFFD0	s5
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
FP	0x7FFFFFC0	
	0x7FFFFFBC	

Frame-based Linkage Convention

Subroutine epilog

```
# restore temporary registers from the stack
mov  $v0,  8($fp)     # return a value
addi $sp,  $fp,  16   # deallocate local vars
lw   $s0,  8($sp)     # pop s0, s3, s5
lw   $s3,  4($sp)
lw   $s5,  0($sp)
addi $sp,  $sp, 12    # deallocate S reg space
lw   $fp,  0($sp)     # pop fp
lw   $ra,  4($sp)     # pop ra
addi $sp,  $sp, 8     # deallocate ra and fp
jr   $ra              # jump register
```

	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
SP	0x7FFFFFDC	fp
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
FP	0x7FFFFFC0	
	0x7FFFFFBC	

Frame-based Linkage Convention

Subroutine epilog

```
# restore temporary registers from the stack
mov  $v0,  8($fp)     # return a value
addi $sp,  $fp,  16   # deallocate local vars
lw   $s0,  8($sp)     # pop s0, s3, s5
lw   $s3,  4($sp)
lw   $s5,  0($sp)
addi $sp,  $sp, 12    # deallocate S reg space
lw   $fp,  0($sp)     # pop fp
lw   $ra,  4($sp)     # pop ra
addi $sp,  $sp, 8     # deallocate ra and fp
jr   $ra              # jump register
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	ra
SP	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	
	0x7FFFFFBC	

Frame-based Linkage Convention

Subroutine epilog

```
# restore temporary registers from the stack
mov  $v0,  8($fp)     # return a value
addi $sp,  $fp,  16   # deallocate local vars
lw   $s0,  8($sp)     # pop s0, s3, s5
lw   $s3,  4($sp)
lw   $s5,  0($sp)
addi $sp,  $sp, 12    # deallocate S reg space
lw   $fp,  0($sp)     # pop fp
lw   $ra,  4($sp)     # pop ra
addi $sp,  $sp, 8     # deallocate ra and fp
jr   $ra              # jump register
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
	0x7FFFFFE4	t3
	0x7FFFFFE0	
SP	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	
	0x7FFFFFBC	

Frame-based Linkage Convention

Subroutine epilog

```
# restore temporary registers from the stack
mov  $v0,  8($fp)     # return a value
addi $sp,  $fp,  16   # deallocate local vars
lw   $s0,  8($sp)     # pop s0, s3, s5
lw   $s3,  4($sp)
lw   $s5,  0($sp)
addi $sp,  $sp, 12    # deallocate S reg space
lw   $fp,  0($sp)     # pop fp
lw   $ra,  4($sp)     # pop ra
addi $sp,  $sp, 8     # deallocate ra and fp
jr   $ra              # jump register
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	t0
SP	0x7FFFFFE4	t3
	0x7FFFFFE0	
	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	
	0x7FFFFFBC	

Frame-based Linkage Convention

Caller restores the control

```
# restore temporary registers from the stack
lw   $t0,  4($sp)     # pop t0, t3
lw   $t3,  0($sp)
addi $sp,  $sp, 8    # deallocate t reg space
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	
SP	0x7FFFFFE4	t3
	0x7FFFFFE0	
	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	
	0x7FFFFFBC	

Frame-based Linkage Convention

Caller restores the control

```
# restore temporary registers from the stack
lw   $t0,  4($sp)     # pop t0, t3
lw   $t3,  0($sp)
addi $sp,  $sp, 8    # deallocate t reg space
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
	0x7FFFFFEC	main a
	0x7FFFFFE8	
SP	0x7FFFFFE4	
	0x7FFFFFE0	
	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	
	0x7FFFFFBC	

Frame-based Linkage Convention

Caller restores the control

```
# restore temporary registers from the stack
lw   $t0,  4($sp)     # pop t0, t3
lw   $t3,  0($sp)
addi $sp,  $sp, 8    # deallocate t reg space
```

FP	0x7FFFFFFC	main y
	0x7FFFFFF8	main x
	0x7FFFFFF4	main c
	0x7FFFFFF0	main b
SP	0x7FFFFFEC	main a
	0x7FFFFFE8	
	0x7FFFFFE4	
	0x7FFFFFE0	
	0x7FFFFFDC	
	0x7FFFFFD8	
	0x7FFFFFD4	
	0x7FFFFFD0	
	0x7FFFFFCC	
	0x7FFFFFC8	
	0x7FFFFFC4	
	0x7FFFFFC0	
	0x7FFFFFBC	