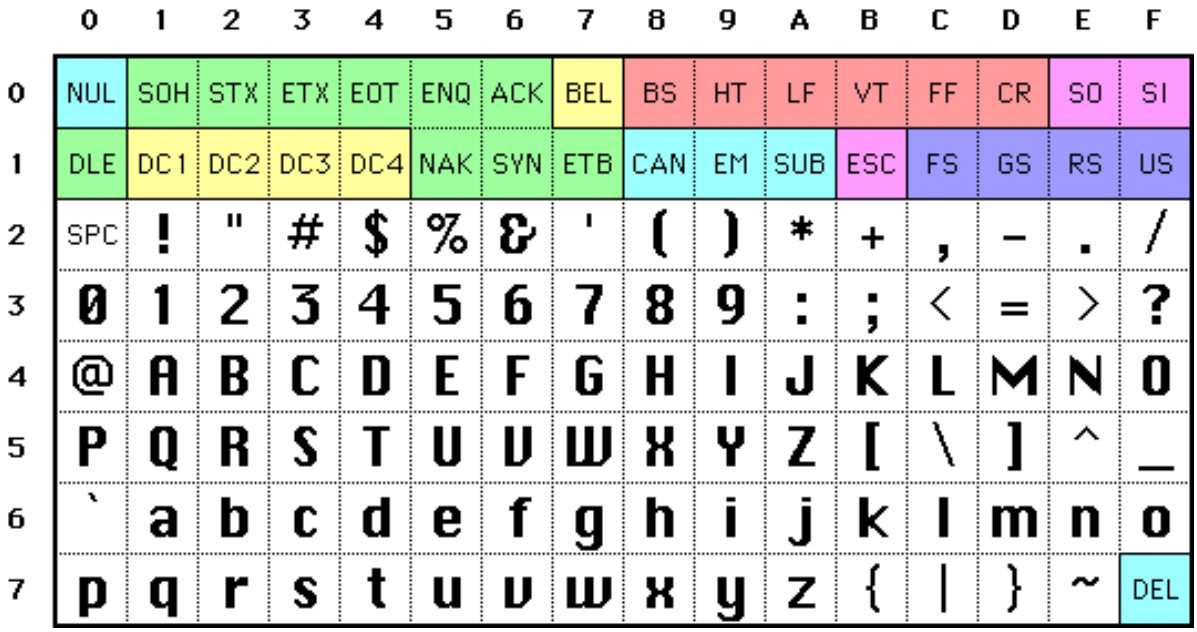


| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | HT | LF | VT | FF | CR | SO | SI |
| 1 | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB | CAN | EM | SUB | ESC | FS | GS | RS | US |
| 2 | SPC | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | DEL |

| Service | Code (in $v0) | Arguments / Result |
|---|---|---|
| print_int | 1 | $a0 = integer value to print |
| print_float | 2 | $f12 = float value to print |
| print_double | 3 | $f12 = double value to print |
| print_string | 4 | $a0 = address of null-terminated string |
| read_int | 5 | $v0 = integer read |
| read_float | 6 | $f0 = float read |
| read_double | 7 | $f0 = double read |
| read_string | 8 | $a0 = buffer address<br>$a1 = buffer size<br>*reads up to "buffer size - 1" characters & null terminates* |
| sbrk | 9 | $a0 = bytes to dynamically allocate<br>$v0 = address of allocated space |
| exit | 10 | |
| print_char | 11 | $a0 = character to print |
| read_char | 12 | $v0 = character read |

| Register Name(s) | Register Number(s) | Intended Use (by convention) | Preserved Across Call? |
|---|---|---|---|
| $zero | 0 | constant value 0 | (N.A.) |
| $at | 1 | reserved for assembler | no |
| $v0, $v1 | 2, 3 | result(s) of procedure | no |
| $a0, …, $a3 | 4, …, 7 | argument(s) of procedure | no |
| $t0, …, $t7 | 8, …, 15 | temporaries | no |
| $s0, …, $s7 | 16, …, 23 | saved temporaries | yes |
| $t8, $t9 | 24, 25 | temporaries | no |
| $k0, $k1 | 26, 27 | reserved for OS kernel | no |
| $gp | 28 | global pointer | yes |
| $sp | 29 | stack pointer | yes |
| $fp | 30 | frame pointer | yes |
| $ra | 31 | return address | yes |

CS 2318 (Assembly Language) **"Course-Relevant" MIPS Instructions/Pseudoinstructions** Lee S. Koh

| | (6) | (5) | (5) | (5) | (5) | (6) |
|---|---|---|---|---|---|---|
| abs Rdest, Rsrc | | | | | | |
| add Rd, Rs, Rt | 0 | Rs | Rt | Rd | 0 | 0x20 |
| addi Rt, Rs, imm | 8 | Rs | Rt | imm | | |
| addiu Rt, Rs, imm | 9 | Rs | Rt | imm | | |
| addu Rd, Rs, Rt | 0 | Rs | Rt | Rd | 0 | 0x21 |
| and Rd, Rs, Rt | 0 | Rs | Rt | Rd | 0 | 0x24 |
| andi Rt, Rs, imm | 0xc | Rs | Rt | imm | | |
| b label | | | | | | |
| beq Rs, Rt, label | 4 | Rs | Rt | WOffset | | |
| beqz Rsrc, label | | | | | | |
| bge Rsrc1, Rsrc2, label | | | | | | |
| bgeu Rsrc1, Rsrc2, label | | | | | | |
| bgez Rs, label | 1 | Rs | 1 | WOffset | | |
| bgt Rsrc1, Rsrc2, label | | | | | | |
| bgtu Rsrc1, Rsrc2, label | | | | | | |
| bgtz Rs, label | 7 | Rs | 0 | WOffset | | |
| ble Rsrc1, Rsrc2, label | | | | | | |
| bleu Rsrc1, Rsrc2, label | | | | | | |
| blez Rs, label | 6 | Rs | 0 | WOffset | | |
| blt Rsrc1, Rsrc2, label | | | | | | |
| bltu Rsrc1, Rsrc2, label | | | | | | |
| bltz Rs, label | 1 | Rs | 0 | WOffset | | |
| bne Rs, Rt, label | 5 | Rs | Rt | WOffset | | |
| bnez Rsrc, label | | | | | | |
| div Rdest, Rsrc1, Rsrc2 | | | | | | |
| div Rs, Rt | 0 | Rs | Rt | 0 | 0 | 0x1a |
| divu Rdest, Rsrc1, Rsrc2 | | | | | | |
| divu Rs, Rt | 0 | Rs | Rt | 0 | 0 | 0x1b |
| j label | 2 | Pseudodirect address | | | | |
| jal label | 3 | Pseudodirect address | | | | |
| jr Rs | 0 | Rs | 0 | 0 | 0 | 8 |
| la Rdest, label | | | | | | |
| lb Rt, BOffset(Rs) | 0x20 | Rs | Rt | BOffset | | |
| lbu Rt, BOffset(Rs) | 0x24 | Rs | Rt | BOffset | | |
| lh Rt, BOffset(Rs) | 0x21 | Rs | Rt | BOffset | | |
| lhu Rt, BOffset(Rs) | 0x25 | Rs | Rt | BOffset | | |
| li Rdest, imm | | | | | | |
| lui Rt, imm | 0xf | 0 | Rt | imm | | |
| lw Rt, BOffset(Rs) | 0x23 | Rs | Rt | BOffset | | |
| mfhi Rd | 0 | 0 | 0 | Rd | 0 | 0x10 |
| mflo Rd | 0 | 0 | 0 | Rd | 0 | 0x12 |
| move Rdest, Rsrc | | | | | | |

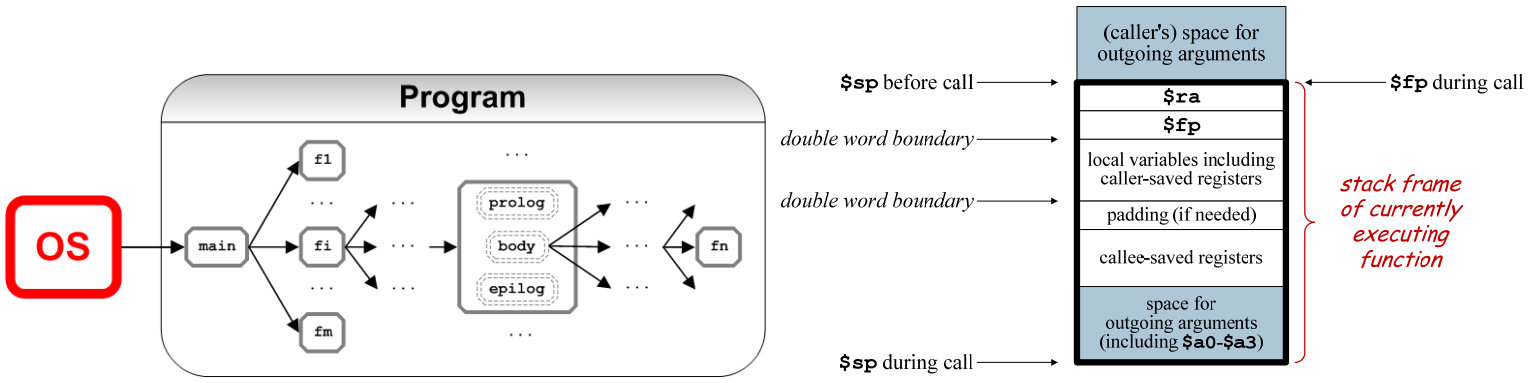
| | |
|---|---|
| 0 | 0 |
| at | 1 |
| v0 | 2 |
| v1 | 3 |
| a0 | 4 |
| a1 | 5 |
| a2 | 6 |
| a3 | 7 |
| t0 | 8 |
| t1 | 9 |
| t2 | 10 |
| t3 | 11 |
| t4 | 12 |
| t5 | 13 |
| t6 | 14 |
| t7 | 15 |
| s0 | 16 |
| s1 | 17 |
| s2 | 18 |
| s3 | 19 |
| s4 | 20 |
| s5 | 21 |
| s6 | 22 |
| s7 | 23 |
| t8 | 24 |
| t9 | 25 |
| k0 | 26 |
| k1 | 27 |
| gp | 28 |
| sp | 29 |
| fp | 30 |
| ra | 31 |

| Opcode | |
|---|---|
| R | 0 |
| I | 1, 4–62 |
| J | 2 or 3 |

| | (6) | (5) | (5) | (5) | (5) | (6) |
|---|---|---|---|---|---|---|
| mthi Rs | 0 | Rs | 0 | 0 | 0 | 0x11 |
| mtlo Rs | 0 | Rs | 0 | 0 | 0 | 0x13 |
| mul Rd, Rs, Rt | 0x1c | Rs | Rt | Rd | 0 | 2 |
| mulo Rdest, Rsrc1, Rsrc2 | | | | | | |
| mulou Rdest, Rsrc1, Rsrc2 | | | | | | |
| mult Rs, Rt | 0 | Rs | Rt | 0 | 0 | 0x18 |
| multu Rs, Rt | 0 | Rs | Rt | 0 | 0 | 0x19 |
| neg Rdest, Rsrc | | | | | | |
| negu Rdest, Rsrc | | | | | | |
| nop | 0 | 0 | 0 | 0 | 0 | 0 |
| nor Rd, Rs, Rt | 0 | Rs | Rt | Rd | 0 | 0x27 |
| not Rdest, Rsrc | | | | | | |
| or Rd, Rs, Rt | 0 | Rs | Rt | Rd | 0 | 0x25 |
| ori Rt, Rs, imm | 0xd | Rs | Rt | imm | | |
| rem Rdest, Rsrc1, Rsrc2 | | | | | | |
| rol Rdest, Rsrc1, Rsrc2 | | | | | | |
| ror Rdest, Rsrc1, Rsrc2 | | | | | | |
| sb Rt, BOffset(Rs) | 0x28 | Rs | Rt | BOffset | | |
| seq Rdest, Rsrc1, Rsrc2 | | | | | | |
| sge Rdest, Rsrc1, Rsrc2 | | | | | | |
| sgeu Rdest, Rsrc1, Rsrc2 | | | | | | |
| sgt Rdest, Rsrc1, Rsrc2 | | | | | | |
| sgtu Rdest, Rsrc1, Rsrc2 | | | | | | |
| sh Rt, BOffset(Rs) | 0x29 | Rs | Rt | BOffset | | |
| sle Rdest, Rsrc1, Rsrc2 | | | | | | |
| sleu Rdest, Rsrc1, Rsrc2 | | | | | | |
| sll Rd, Rt, shamt | 0 | 0 | Rt | Rd | shamt | 0 |
| sllv Rd, Rt, Rs | 0 | Rs | Rt | Rd | 0 | 4 |
| slt Rd, Rs, Rt | 0 | Rs | Rt | Rd | 0 | 0x2a |
| slti Rt, Rs, imm | 0xa | Rs | Rt | imm | | |
| sltiu Rt, Rs, imm | 0xb | Rs | Rt | imm | | |
| sltu Rd, Rs, Rt | 0 | Rs | Rt | Rd | 0 | 0x2b |
| sne Rdest, Rsrc1, Rsrc2 | | | | | | |
| sra Rd, Rt, shamt | 0 | 0 | Rt | Rd | shamt | 3 |
| srav Rd, Rt, Rs | 0 | Rs | Rt | Rd | 0 | 7 |
| srl Rd, Rt, shamt | 0 | 0 | Rt | Rd | shamt | 2 |
| srlv Rd, Rt, Rs | 0 | Rs | Rt | Rd | 0 | 6 |
| sub Rd, Rs, Rt | 0 | Rs | Rt | Rd | 0 | 0x22 |
| subu Rd, Rs, Rt | 0 | Rs | Rt | Rd | 0 | 0x23 |
| sw Rt, BOffset(Rs) | 0x2b | Rs | Rt | BOffset | | |
| xor Rd, Rs, Rt | 0 | Rs | Rt | Rd | 0 | 0x26 |
| xori Rt, Rs, imm | 0xe | Rs | Rt | imm | | |

# MIPS Function-call Convention Used in This Course

(for "***non-trivial***" functions each ***requiring a stack frame*** and involving ***non-floating-point*** data)




■ Callee

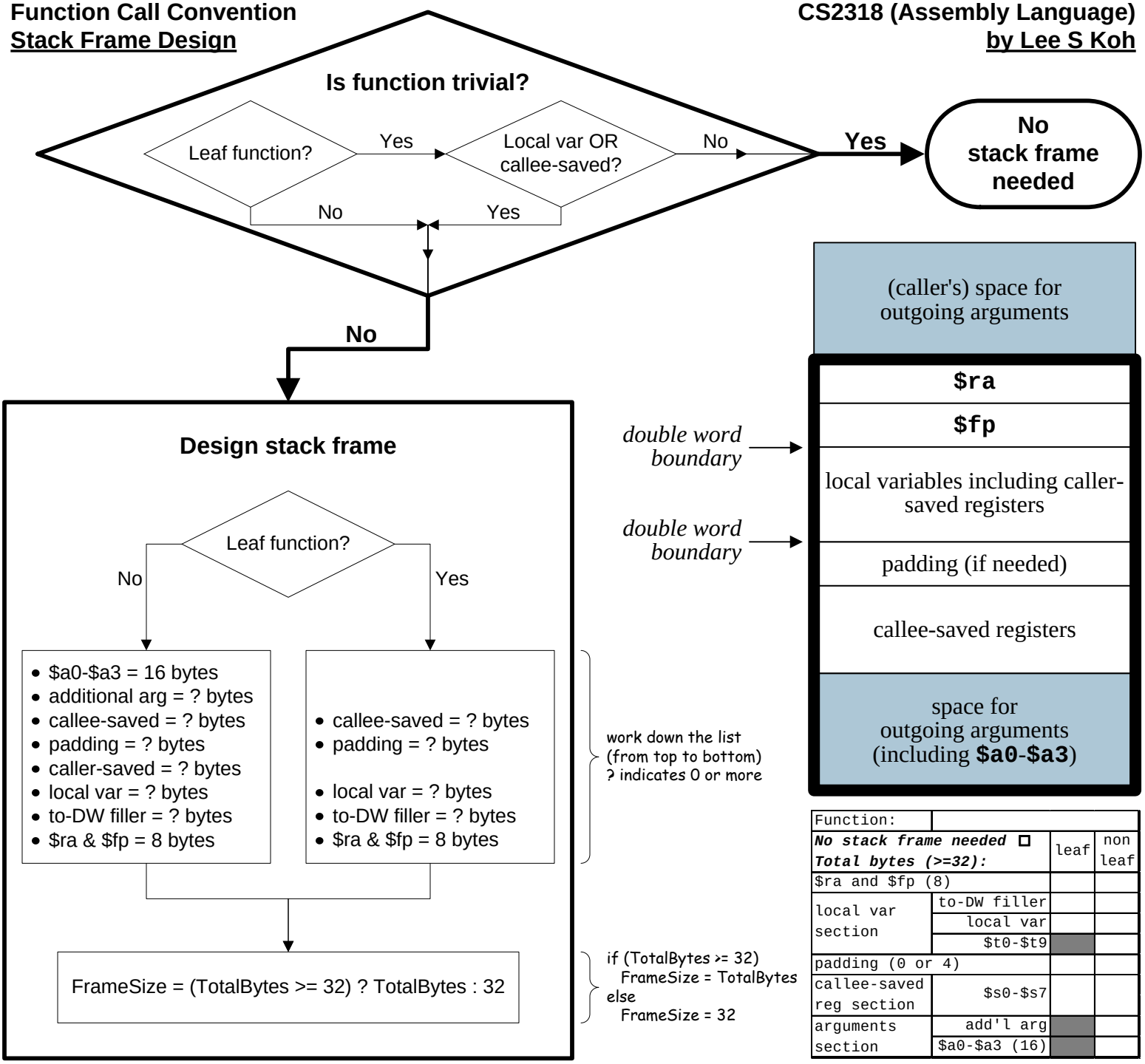
- *prolog* or *preamble*
  - construct stack frame with frame size determined as follows:
    - *always* have frame size *in bytes* that is *at least 32* AND *divisible by (multiple of) 8*
    - *always* allocate *24 bytes* for storing the 6 registers **`$ra`**, **`$fp`** and **`$a0-$a3`**
    - *where appropriate* allocate for *callee-saved registers*
    - *where appropriate* allocate for *caller-saved registers*
    - *where appropriate* allocate for *local variables*
    - *where appropriate* allocate for *additional outgoing arguments* (beyond the 4 that's always allocated)
  - save contents of *calle-saved registers*
    - *always* save **`$ra`** and **`$fp`**
    - *where appropriate* save any *callee-saved* registers
  - set **`$fp`** to "old" value of **`$sp`**
- *body*
  - Before calling a function
    - save any *caller-saved* registers
    - place any 5th and beyond arguments on stack
      - » place any 1st through 4th arguments in registers **`$a0-$a3`**
  - Immediately after a called function returns
    - restore any (previously saved) *caller-saved* registers
- *epilog*
  - place any return values
  - restore any (previously saved) callee-saved registers
  - tear down stack frame
  - return to caller

| Function: | | | |
|---|---|---|---|
| ***No stack frame needed*** ☐ ***Total bytes (>=32):*** | | leaf | non leaf |
| $ra and $fp (8) | | | |
| local var section | to-DW filler | | |
| | local var | | |
| | $t0-$t9 | | |
| padding (0 or 4) | | | |
| callee-saved reg section | $s0-$s7 | | |
| arguments section | add'l arg | | |
| | $a0-$a3 (16) | | |


| Function: | | | |
|---|---|---|---|
| ***No stack frame needed*** ☐ ***Total bytes (>=32):*** | | leaf | non leaf |
| $ra and $fp (8) | | | |
| local var section | to-DW filler | | |
| | local var | | |
| | $t0-$t9 | | |
| padding (0 or 4) | | | |
| callee-saved reg section | $s0-$s7 | | |
| arguments section | add'l arg | | |
| | $a0-$a3 (16) | | |


■ Caller (a function plays the role of caller whenever it makes a call)

- *pre-call*
  - *where appropriate* save any *caller-saved* registers
  - *where appropriate* place first 4 arguments in **`$a0-$a3`** and 5th and beyond arguments on stack
- *post-call*
  - *where appropriate* restore any (previously saved) *caller-saved* registers
  - *where appropriate* retrieve *return values*




| Function: | | | |
|---|---|---|---|
| ***No stack frame needed ☐ Total bytes (>=32):*** | | leaf | non leaf |
| $ra and $fp (8) | | | |
| local var section | to-DW filler | | |
| | local var | | |
| | $t0-$t9 | | |
| padding (0 or 4) | | | |
| callee-saved reg section | $s0-$s7 | | |
| arguments section | add'l arg | | |
| | $a0-$a3 (16) | | |

**Notes:**

- A *non-leaf* function always requires a stack frame (*i.e.*, is always *non-trivial*).
- A *leaf* function requires a stack frame only if (*i.e.*, is *non-trivial* only if) one or more of the following apply:
  - It needs *local storage* (variables that may include arrays) on the stack when performing its task.
  - It wants/needs to use *calle-saved registers* (**`$s0-$s7`**) when performing its task.
- For a *leaf* function that requires a stack frame (*i.e.*, one that is *non-trivial*):
  - Contents of **`$ra`** and **`$fp`** must still be saved (in the *prolog*) and restored (in the *epilog*).
  - Therefore, space for saving **`$ra`** and **`$fp`** must be included in the calculation of frame size.
  - If *callee-saved registers section* applies, must still consider whether there is need for "padding".
  - If *local-variable section* applies, must still consider whether there is need for "to-DW filler".
  - Minimum frame size is still 32 bytes (8 words).