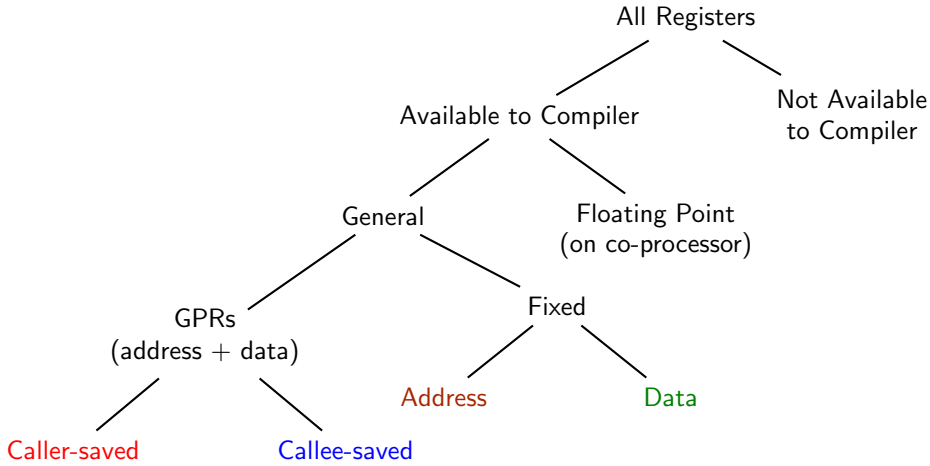
Registers Usage in sclp

Register Categories for Spim



Registers in Spim

Name	No.	Sizes	Use	Category
$zero	00	32		constant data
at	01	32		NA
v0	02	32,64	expr result	caller-saved
v1	03	32	function result	caller-saved
a0	04	32,64	argument	caller-saved
a1	05	32	argument	caller-saved
a2	06	32,64	argument	caller-saved
a3	07	32	argument	caller-saved
t0	08	32,64	temporary	caller-saved
t1	09	32	temporary	caller-saved
t2	10	32,64	temporary	caller-saved
t3	11	32	temporary	caller-saved
t4	12	32,64	temporary	caller-saved
t5	13	32	temporary	caller-saved
t6	14	32,64	temporary	caller-saved
t7	15	32	temporary	caller-saved

Name	No.	Sizes	Use	Category
s0	16	32,64	temporary	callee-saved
s1	17	32	temporary	callee-saved
s2	18	32,64	result	callee-saved
s3	19	32	result	callee-saved
s4	20	32,64	temporary	callee-saved
s5	21	32	temporary	callee-saved
s6	22	32,64	temporary	callee-saved
s7	23	32	temporary	callee-saved
t8	24	32,64	temporary	caller-saved
t9	25	32	temporary	caller-saved
k0	26	32,64		NA
k1	27	32		NA
gp	28	32,64	global pointer	address
sp	29	32	stack pointer	address
fp	30	32,64	frame pointer	address
ra	31	32	return address	address

Co-Processor Registers in Spim

Name	Number	Sizes
f0	00	32,64
f1	01	32
f2	02	32,64
f3	03	32
f4	04	32,64
f5	05	32
f6	06	32,64
f7	07	32
f8	08	32,64
f9	09	32
f10	10	32,64
f11	11	32
f12	12	32,64
f13	13	32
f14	14	32,64
f15	15	32

Name	Number	Sizes
f16	16	32,64
f17	17	32
f19	18	32,64
f19	19	32
f20	20	32,64
f21	21	32
f22	22	32,64
f23	23	32
f24	24	32,64
f25	25	32
f26	26	32,64
f27	27	32
f28	28	32,64
f29	29	32
f30	30	32,64
f31	31	32

Registers Used By SCLP for Storing Intermediate Results

Register		Type
Name	Pair	
v0		int
v1		int
t0		int
t1		int
t2		int
t3		int
t4		int
t5		int
t6		int
t7		int
t8		int
t9		int

Register		Type
Name	Pair	
s0		int
s1		int
s2		int
s3		int
s4		int
s5		int
s6		int
s7		int
f0	f0,f1	float
f2	f2,f3	float
f4	f4,f5	float
f6	f5,f6	float

Register		Type
Name	Pair	
f8	f8,f9	float
f10	f10,f11	float
f12	f12,f13	float
f14	f14,f15	float
f16	f16,f17	float
f18	f18,f19	float
f20	f20,f21	float
f22	f22,f23	float
f24	f24,f25	float
f26	f26,f27	float
f28	f28,f29	float
f30	f30,f31	float

Registers Used By SCLP for Storing Intermediate Results

Designated for function result, hence ignored

Register		Type
Name	Pair	
v0		int
v1		int
t0		int
t1		int
t2		int
t3		int
t4		int
t5		int
t6		int
t7		int
t8		int
t9		int

Register		Type
Name	Pair	
s0		int
s1		int
s2		int
s3		int
s4		int
s5		int
s6		int
s7		int
f0	f0,f1	float
f2	f2,f3	float
f4	f4,f5	float
f6	f5,f6	float

Register		Type
Name	Pair	
f8	f8,f9	float
f10	f10,f11	float
f12	f12,f13	float
f14	f14,f15	float
f16	f16,f17	float
f18	f18,f19	float
f20	f20,f21	float
f22	f22,f23	float
f24	f24,f25	float
f26	f26,f27	float
f28	f28,f29	float
f30	f30,f31	float

Registers Used By SCLP for Storing Intermediate Results

Designated for function result, hence ignored

Register		Type
Name	Pair	
v0		int
t0		int
t1		int
t2		int
t3		int
t4		int
t5		int
t6		int
t7		int
t8		int
t9		int

Register		Type
Name	Pair	
s0		int
s1		int
s2		int
s3		int
s4		int
s5		int
s6		int
s7		int
f2	f2,f3	float
f4	f4,f5	float
f6	f5,f6	float

Register		Type
Name	Pair	
f8	f8,f9	float
f10	f10,f11	float
f12	f12,f13	float
f14	f14,f15	float
f16	f16,f17	float
f18	f18,f19	float
f20	f20,f21	float
f22	f22,f23	float
f24	f24,f25	float
f26	f26,f27	float
f28	f28,f29	float
f30	f30,f31	float

Register Allocation Policy Used by sclp

Very simple strategy

- No register is occupied across any assignment in the source program

  The result is stored in memory for the LHS variable and the result register is freed
- Within an expression, values of source variables are loaded into registers
- Intermediate values within an expression (stored in a temporary variable) are assigned a register

  The result of a ternary expression is a "saved" temporary which is treated like a source variable
- Getting a new register
  - When a temporary is assigned a register, mark it as occupied
  - When a temporary is used in a TAC statement, mark the register as free
  - To get a new register,
    - Traverse the list of registers and assign the first free register
    - Need to match the type
- Registers are chosen for a TAC statement in this order: first operand, result, second operand (because second operand may not exist)