### Register Allocation

- How to best use the bounded number of registers.
- Use or registers
  - Register allocation
    - We select a set of variables that will reside in registers at each point in the program
  - Register assignment
    - We pick the specific register that a variable will reside in.
- Complications:
  - special purpose registers
  - operators requiring multiple registers.
- · Optimal assignment is NP-complete

# Register Allocation

le:							
t	:=	a	+	b			
t	:=	t	*	С			
t	:=	t	/	d			
Target Code:							
mov		a,r1					
add		b,r1					
mul		c,r0					
div		d,r0					
mov		r1,t					
	t t t t Coo mo accomu	t := t := t := Code: mov add mul div	t := a t := t t := t  Code: mov a add b mul c div c	t := a + t := t * t := t / Code: mov a,: add b,: mul c,: div d,:			

IR Code	:					
t	: :=	a	+	b		
t	: :=	t	+	C		
l t	: :=	t	/	d		
Target C	ode:					
m	mov		a,r0			
a	add		b,r0			
a	add		c,r0			
s	srda		32,r0			
d	div		d,r0			
m	mov		r1,t			

#### **Conclusion:**

Where you put the result of t:=a+b (either r0 or r1) depends on how it will be used later!!!

[A "chicken-and-egg" problem]

### Register Allocation

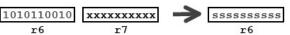
Multiply Instruction

div 
$$y, r4 \leftarrow$$
 Must specify an even numbered register  $[r4.r5] \div y \Rightarrow [r4.r5]$ 

1010110010

r7

SRDA: Shift Right Double Arithmetic



#### Instruction Scheduling

- Choosing the order of instructions to best utilize resources
- Picking the optimal order is NP-complete problem
- Simplest Approach
  - Don't mess with re-ordering.
  - Target code will perform all operations in the same order as the IR code
- Trickier Approach
  - Consider re-ordering operations
  - May produce better code
    - ... Get operands into registers just before they are needed
    - ... May use registers more efficiently

## Moving Results Back to Memory

- When to move results from registers back into memory?
  - After an operation, the result will be in a register.
- Immediately
  - Move data back to memory just after it is computed.
  - May make more registers available for use elsewhere.
- Wait as long as possible before moving it back
  - Only move data back to memory "at the end"
    - or "when absolutely necessary"
  - May be able to avoid re-loading it later!