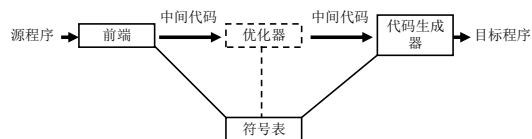


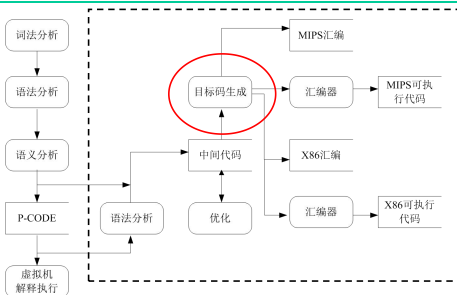
## 第十五章 目标代码生成

面向目标体系结构的代码生成和优化技术

## 代码生成器在编译系统中的位置



## 教学编译器架构



## 代码生成器的输入

- 源程序的中间表示
  - 线性表示（波兰式）
  - 三地址码（四元式）
  - 栈式中间代码（P-CODE/Java Bytecode）
  - 图形表示
- 符号表信息

## 代码生成器对输入的要求

- 编译器前端已经将源程序扫描、分析和翻译成足够详细的**中间表示**
- 中间语言中的**标识符**表示为目标机器能够直接操作的变量（位、整数、浮点数、指针等）
- 完成了必要的类型检查，类型转换/检测操作已经加入到中间语言的必要位置
- 完成语法和必要的语义检查**，代码生成器可以认为输入中没有与语法或语义错误

## 目标程序的种类

- 汇编语言
  - 生成宏汇编代码，再由汇编程序进行编译，连接，从而生成最终代码（.S/.ASM文件）
- 包含绝对地址的机器语言
  - 执行时必须被载入到地址空间中（相对）固定的位置
  - EXE (MS-WIN)、COM (MS-WIN)、A.OUT (Linux)
- 可重定位的机器语言
  - 一组可重定位的模块子程序可以用连接器装配后生成最终的目标程序（.obj/.o文件组）
  - 可动态加载的模块子程序（DLL/.SO动态连接库）

## 面向特定的目标体系结构生成目标代码

- 目标体系结构可以是：
  - 某种微处理器，如X86、MIPS、ARM等
  - 某种被精心设计和定义的虚拟机或运行时系统，如Java虚拟机、C#运行时系统、P-code虚拟机等。
- 虚拟机：
  - 第十章介绍的面向P-code虚拟机，采用自顶向下的属性翻译文法生成代码的方法适用于其它虚拟机
  - 虚拟机的代码需要解释器解释或者即时编译器编译后才能运行

## 本章内容

### 面向微处理器体系结构的代码生成技术

- 主要内容：
  - 目标代码地址空间的划分，目标体系结构上存贮单元（如寄存器和内存单元）的分配和指派
  - 从中间代码（或者源代码）到目标代码转换过程中所进行的指令选择
  - 面向目标体系结构的优化

## 12.1 现代微处理器体系结构简介

- 指令集
  - Instruction Set
- 流水线和指令级并行
  - Pipeline and Instruction Level Parallelism
- 存储结构和I/O
  - Memory Hierarchy and I/O Systems
- 多处理器和线程级并行
  - Multiprocessor and Thread Level Parallelism

## 12.1.1 指令集架构

不同：算术逻辑单元ALU对存储单元的访问方式不同

## 1、栈式指令集架构

类似于P-code和Java虚拟机
 

$C=A+B$ 
 代码：
 

```
PUSH A
PUSH B
ADD
POP C
```

## 2、累加器式指令集架构

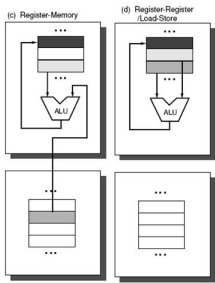
$C=A+B$ 
 代码：
 

```
LOAD A
ADD B
STORE C
```

代码短了，开销不一定小。  
直接访问内容

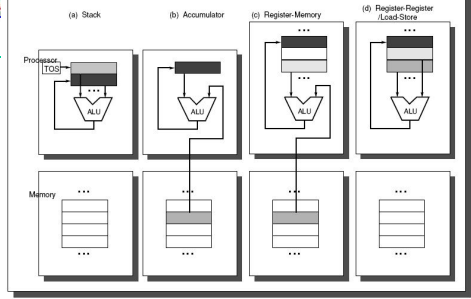
2

寄存器架构



北京航空航天大学计算机学院

- 寄存器-内存指令集架构
- 寄存器-寄存器指令架构
- 区别: 是否可以直接内存寻址
- 共性: 内部有多个寄存器可以直接作为ALU指令的任一操作数
- 优点:
  - 减少内存访问
  - 减少指令数



北京航空

Stack	Accumulator	Register (register-memory)	Register (load-store)
Push A	Load A	Load R1, A	Load R1, A
Push B	Add B	Add R3, R1, B	Load R2, B
Add	Store C	Store R3, C	Add R3, R1, R2
Pop C			Store R3, C

Excellence in  
BUAA SET

$$D=(A*B)+(B*C)$$

- 栈式架构
  - PUSH A
  - PUSH B
  - MUL
  - PUSH B
  - PUSH C
  - MUL
  - ADD
  - POP D
- 寄存器-寄存器架构
  - LOAD R1, A
  - LOAD R2, B
  - LOAD R3, C
  - MUL R1, R2, R4
  - MUL R2, R3, R5
  - ADD R4, R5, R5
  - STORE R5, D

8条指令, 5条内存访问  
北京航空航天大学计算机学院

7条指令, 4条存储访问

Excellence in  
BUAA SET

- 寄存器-内存指令集架构处理器: 称为Complex Instruction Set Computers (CISC)架构计算机
  - 包括使用最广泛的Intel X86架构处理器、曾经十分辉煌, 但现在已经退出历史舞台的DEC VAX系列计算机。
- 寄存器-寄存器指令集架构处理器: 称为Reduced Instruction Set Computer (RISC)架构计算机
  - 包括正在广泛使用的Alpha、ARM、MIPS、PowerPC、SPARC等微处理器。

北京航空航天大学计算机学院

Excellence in  
BUAA SET

12.1.2 存储层次架构

寄存器、缓存、内存、硬盘的存储访问特性



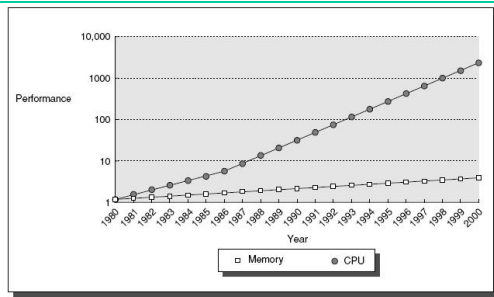
Level	1	2	3	4
Called	Registers	Cache	Main memory	Disk storage
Typical size	< 1 KB	< 16 MB	< 16 GB	> 100 GB
Implementation technology	Custom memory with multiple ports, CMOS, SRAM	On-chip or off-chip CMOS, SRAM	CMOS DRAM	Magnetic disk
Access time (in ns)	0.25-0.5	0.5 to 25	80-250	5,000,000
Bandwidth (in MB/sec)	20,000-100,000	5,000-10,000	1,000-3,000	20-150
Managed by	Compiler	Hardware	Operating system	Operating system/operator
Backed by	Cache	Main memory	Disk	CD or Tape

寄存器的访问速度基本可以保证处理器在每个时钟周期内访问到需要的数据

北京航空航天大学计算机学院

Excellence in  
BUAA SET

1980~2000年, CPU性能和内存访问性能的提高



北京航空航天大学计算机学院

Excellence in  
BUAA SET

## 尽可能少地访问寄存器之外的存储设备

- 但是，寄存器的数量极其有限
  - 32位X86微处理器上有8个通用寄存器
  - X86/ARM, MIPS: 大约16~32个通用寄存器
  - 分配策略很重要。
- 对缓存的利用，对于大型数据结构有用
  - 缓存的管理单位是：缓存行。（数十或上百字节）
  - 每次从内存载入的是一组地址连续的数据，而不仅仅是被访问数据

通过循环交换（Loop Interchange）优化提高缓存命中率

```
for (j = 0; j < 100; j = j+1)
    for (i = 0; i < 5000; i = i+1)
        x[i][j] = 2 * x[i][j];
```

C语言存储：行优先

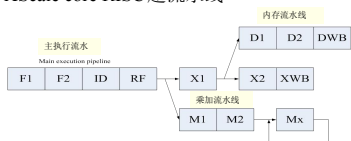
x[0][0], x[0][1], ..., x[0][99], x[1][0], x[1][1], ..., x[1][99], ..., x[4999][0], x[4999][1], ..., x[4999][99]

```
for (i = 0; i < 5000; i = i+1)
    for (j = 0; j < 100; j = j+1)
        x[i][j] = 2 * x[i][j];
```

x[0][0], x[0][1], ..., x[0][99], x[1][0], x[1][1], ..., x[1][99], ..., x[4999][0], x[4999][1], ..., x[4999][99]

## 12.1.3 流水线

XScale core RISC超流水线



10: add R0, R5, R6	1	12: ldr R2, [R4, 0x4]	3*
11: sub R1, R7, R8	1	10: add R0, R5, R6	1
12: ldr R2, [R4, 0x4]	3	11: sub R1, R7, R8	1
13: add R3, R2, R1	1	13: add R3, R2, R1	1

时钟周期: 1+1+3+1 = 6      时钟周期: 1+1+1+1 = 4

- 实现指令级的并行。
- 详见教材p 283

## 12.2 地址空间

- 代码区
  - 存放目标代码
- 静态数据区
  - 全局变量
  - 静态变量
  - 部分常量，例如字符串
- 动态内存区
  - 也被称为内存堆Heap
  - 程序员管理：C、C++
  - 自动管理（内存垃圾收集器）：Java、Ada
- 程序运行栈
  - 活动记录
  - 函数调用的上下文现场
    - 由调用方保存的一些临时寄存器
    - 被调用方保存的一些全局寄存器

## 12.2.1 程序地址空间的实例分析

```
1 // C12P1.cpp
2 #include "stdafx.h"
3 int global_val = 0;
4 int foo(int n);
5 static int static_val = 0;
6 int i;
7 for(i=0; i<100; i++){
8     n = n * i + global_val + static_val;
9 }
10 static_val = n / 2;
11 printf("static_val is %x\n", static_val);
12 return n;
13 }
14 int main(int argc, char* argv[]){
15     global_val = 99;
16     int n = foo(100);
17     printf("foo(100) is %x\n", n);
18     return 0;
19 }
```

```
1 TITLE C12P1.cpp
2 .386P
3 .model FLAT
4
5 PUBLIC ?global_val@3HA
6 ?global_val@3HA
7 ?foo@3HA
8 ?foo@3HA
9 ?foo@3HA
10 ?foo@3HA
11 ?foo@3HA
12 ?foo@3HA
13 ?foo@3HA
14 ?foo@3HA
15 ?foo@3HA
16 ?foo@3HA
17 ?foo@3HA
18 ?foo@3HA
19 ?foo@3HA
20 ?foo@3HA
21 ?foo@3HA
22 ?foo@3HA
23 ?foo@3HA
24 ?foo@3HA
25 ?foo@3HA
26 ?foo@3HA
27 ?foo@3HA
28 ?foo@3HA
29 ?foo@3HA
30 ?foo@3HA
31 ?foo@3HA
32 ?foo@3HA
33 ?foo@3HA
34 ?foo@3HA
35 ?foo@3HA
36 ?foo@3HA
37 ?foo@3HA
38 ?foo@3HA
39 ?foo@3HA
40 ?foo@3HA
41 ?foo@3HA
42 ?foo@3HA
43 ?foo@3HA
44 ?foo@3HA
45 ?foo@3HA
46 ?foo@3HA
47 ?foo@3HA
48 ?foo@3HA
49 ?foo@3HA
50 ?foo@3HA
51 ?foo@3HA
52 ?foo@3HA
53 ?foo@3HA
54 ?foo@3HA
55 ?foo@3HA
56 ?foo@3HA
57 ?foo@3HA
58 ?foo@3HA
59 ?foo@3HA
60 ?foo@3HA
61 ?foo@3HA
62 ?foo@3HA
63 ?foo@3HA
64 ?foo@3HA
65 ?foo@3HA
66 ?foo@3HA
67 ?foo@3HA
68 ?foo@3HA
69 ?foo@3HA
70 ?foo@3HA
71 ?foo@3HA
72 ?foo@3HA
73 ?foo@3HA
74 ?foo@3HA
75 ?foo@3HA
76 ?foo@3HA
77 ?foo@3HA
78 ?foo@3HA
79 ?foo@3HA
80 ?foo@3HA
81 ?foo@3HA
82 ?foo@3HA
83 ?foo@3HA
84 ?foo@3HA
85 ?foo@3HA
86 ?foo@3HA
87 ?foo@3HA
88 ?foo@3HA
89 ?foo@3HA
90 ?foo@3HA
91 ?foo@3HA
92 ?foo@3HA
93 ?foo@3HA
94 ?foo@3HA
95 ?foo@3HA
96 ?foo@3HA
97 ?foo@3HA
98 ?foo@3HA
99 ?foo@3HA
100 ?foo@3HA
101 ?foo@3HA
102 ?foo@3HA
103 ?foo@3HA
104 ?foo@3HA
105 ?foo@3HA
106 ?foo@3HA
107 ?foo@3HA
108 ?foo@3HA
109 ?foo@3HA
110 ?foo@3HA
111 ?foo@3HA
112 ?foo@3HA
113 ?foo@3HA
114 ?foo@3HA
115 ?foo@3HA
116 ?foo@3HA
117 ?foo@3HA
118 ?foo@3HA
119 ?foo@3HA
120 ?foo@3HA
121 ?foo@3HA
122 ?foo@3HA
123 ?foo@3HA
124 ?foo@3HA
125 ?foo@3HA
126 ?foo@3HA
127 ?foo@3HA
128 ?foo@3HA
129 ?foo@3HA
130 ?foo@3HA
131 ?foo@3HA
132 ?foo@3HA
133 ?foo@3HA
134 ?foo@3HA
135 ?foo@3HA
136 ?foo@3HA
137 ?foo@3HA
138 ?foo@3HA
139 ?foo@3HA
140 ?foo@3HA
141 ?foo@3HA
142 ?foo@3HA
143 ?foo@3HA
144 ?foo@3HA
145 ?foo@3HA
146 ?foo@3HA
147 ?foo@3HA
148 ?foo@3HA
149 ?foo@3HA
150 ?foo@3HA
151 ?foo@3HA
152 ?foo@3HA
153 ?foo@3HA
154 ?foo@3HA
155 ?foo@3HA
156 ?foo@3HA
157 ?foo@3HA
158 ?foo@3HA
159 ?foo@3HA
160 ?foo@3HA
161 ?foo@3HA
162 ?foo@3HA
163 ?foo@3HA
164 ?foo@3HA
165 ?foo@3HA
166 ?foo@3HA
167 ?foo@3HA
168 ?foo@3HA
169 ?foo@3HA
170 ?foo@3HA
171 ?foo@3HA
172 ?foo@3HA
173 ?foo@3HA
174 ?foo@3HA
175 ?foo@3HA
176 ?foo@3HA
177 ?foo@3HA
178 ?foo@3HA
179 ?foo@3HA
180 ?foo@3HA
181 ?foo@3HA
182 ?foo@3HA
183 ?foo@3HA
184 ?foo@3HA
185 ?foo@3HA
186 ?foo@3HA
187 ?foo@3HA
188 ?foo@3HA
189 ?foo@3HA
190 ?foo@3HA
191 ?foo@3HA
192 ?foo@3HA
193 ?foo@3HA
194 ?foo@3HA
195 ?foo@3HA
196 ?foo@3HA
197 ?foo@3HA
198 ?foo@3HA
199 ?foo@3HA
200 ?foo@3HA
201 ?foo@3HA
202 ?foo@3HA
203 ?foo@3HA
204 ?foo@3HA
205 ?foo@3HA
206 ?foo@3HA
207 ?foo@3HA
208 ?foo@3HA
209 ?foo@3HA
210 ?foo@3HA
211 ?foo@3HA
212 ?foo@3HA
213 ?foo@3HA
214 ?foo@3HA
215 ?foo@3HA
216 ?foo@3HA
217 ?foo@3HA
218 ?foo@3HA
219 ?foo@3HA
220 ?foo@3HA
221 ?foo@3HA
222 ?foo@3HA
223 ?foo@3HA
224 ?foo@3HA
225 ?foo@3HA
226 ?foo@3HA
227 ?foo@3HA
228 ?foo@3HA
229 ?foo@3HA
230 ?foo@3HA
231 ?foo@3HA
232 ?foo@3HA
233 ?foo@3HA
234 ?foo@3HA
235 ?foo@3HA
236 ?foo@3HA
237 ?foo@3HA
238 ?foo@3HA
239 ?foo@3HA
240 ?foo@3HA
241 ?foo@3HA
242 ?foo@3HA
243 ?foo@3HA
244 ?foo@3HA
245 ?foo@3HA
246 ?foo@3HA
247 ?foo@3HA
248 ?foo@3HA
249 ?foo@3HA
250 ?foo@3HA
251 ?foo@3HA
252 ?foo@3HA
253 ?foo@3HA
254 ?foo@3HA
255 ?foo@3HA
256 ?foo@3HA
257 ?foo@3HA
258 ?foo@3HA
259 ?foo@3HA
260 ?foo@3HA
261 ?foo@3HA
262 ?foo@3HA
263 ?foo@3HA
264 ?foo@3HA
265 ?foo@3HA
266 ?foo@3HA
267 ?foo@3HA
268 ?foo@3HA
269 ?foo@3HA
270 ?foo@3HA
271 ?foo@3HA
272 ?foo@3HA
273 ?foo@3HA
274 ?foo@3HA
275 ?foo@3HA
276 ?foo@3HA
277 ?foo@3HA
278 ?foo@3HA
279 ?foo@3HA
280 ?foo@3HA
281 ?foo@3HA
282 ?foo@3HA
283 ?foo@3HA
284 ?foo@3HA
285 ?foo@3HA
286 ?foo@3HA
287 ?foo@3HA
288 ?foo@3HA
289 ?foo@3HA
290 ?foo@3HA
291 ?foo@3HA
292 ?foo@3HA
293 ?foo@3HA
294 ?foo@3HA
295 ?foo@3HA
296 ?foo@3HA
297 ?foo@3HA
298 ?foo@3HA
299 ?foo@3HA
300 ?foo@3HA
301 ?foo@3HA
302 ?foo@3HA
303 ?foo@3HA
304 ?foo@3HA
305 ?foo@3HA
306 ?foo@3HA
307 ?foo@3HA
308 ?foo@3HA
309 ?foo@3HA
310 ?foo@3HA
311 ?foo@3HA
312 ?foo@3HA
313 ?foo@3HA
314 ?foo@3HA
315 ?foo@3HA
316 ?foo@3HA
317 ?foo@3HA
318 ?foo@3HA
319 ?foo@3HA
320 ?foo@3HA
321 ?foo@3HA
322 ?foo@3HA
323 ?foo@3HA
324 ?foo@3HA
325 ?foo@3HA
326 ?foo@3HA
327 ?foo@3HA
328 ?foo@3HA
329 ?foo@3HA
330 ?foo@3HA
331 ?foo@3HA
332 ?foo@3HA
333 ?foo@3HA
334 ?foo@3HA
335 ?foo@3HA
336 ?foo@3HA
337 ?foo@3HA
338 ?foo@3HA
339 ?foo@3HA
340 ?foo@3HA
341 ?foo@3HA
342 ?foo@3HA
343 ?foo@3HA
344 ?foo@3HA
345 ?foo@3HA
346 ?foo@3HA
347 ?foo@3HA
348 ?foo@3HA
349 ?foo@3HA
350 ?foo@3HA
351 ?foo@3HA
352 ?foo@3HA
353 ?foo@3HA
354 ?foo@3HA
355 ?foo@3HA
356 ?foo@3HA
357 ?foo@3HA
358 ?foo@3HA
359 ?foo@3HA
360 ?foo@3HA
361 ?foo@3HA
362 ?foo@3HA
363 ?foo@3HA
364 ?foo@3HA
365 ?foo@3HA
366 ?foo@3HA
367 ?foo@3HA
368 ?foo@3HA
369 ?foo@3HA
370 ?foo@3HA
371 ?foo@3HA
372 ?foo@3HA
373 ?foo@3HA
374 ?foo@3HA
375 ?foo@3HA
376 ?foo@3HA
377 ?foo@3HA
378 ?foo@3HA
379 ?foo@3HA
380 ?foo@3HA
381 ?foo@3HA
382 ?foo@3HA
383 ?foo@3HA
384 ?foo@3HA
385 ?foo@3HA
386 ?foo@3HA
387 ?foo@3HA
388 ?foo@3HA
389 ?foo@3HA
390 ?foo@3HA
391 ?foo@3HA
392 ?foo@3HA
393 ?foo@3HA
394 ?foo@3HA
395 ?foo@3HA
396 ?foo@3HA
397 ?foo@3HA
398 ?foo@3HA
399 ?foo@3HA
400 ?foo@3HA
401 ?foo@3HA
402 ?foo@3HA
403 ?foo@3HA
404 ?foo@3HA
405 ?foo@3HA
406 ?foo@3HA
407 ?foo@3HA
408 ?foo@3HA
409 ?foo@3HA
410 ?foo@3HA
411 ?foo@3HA
412 ?foo@3HA
413 ?foo@3HA
414 ?foo@3HA
415 ?foo@3HA
416 ?foo@3HA
417 ?foo@3HA
418 ?foo@3HA
419 ?foo@3HA
420 ?foo@3HA
421 ?foo@3HA
422 ?foo@3HA
423 ?foo@3HA
424 ?foo@3HA
425 ?foo@3HA
426 ?foo@3HA
427 ?foo@3HA
428 ?foo@3HA
429 ?foo@3HA
430 ?foo@3HA
431 ?foo@3HA
432 ?foo@3HA
433 ?foo@3HA
434 ?foo@3HA
435 ?foo@3HA
436 ?foo@3HA
437 ?foo@3HA
438 ?foo@3HA
439 ?foo@3HA
440 ?foo@3HA
441 ?foo@3HA
442 ?foo@3HA
443 ?foo@3HA
444 ?foo@3HA
445 ?foo@3HA
446 ?foo@3HA
447 ?foo@3HA
448 ?foo@3HA
449 ?foo@3HA
450 ?foo@3HA
451 ?foo@3HA
452 ?foo@3HA
453 ?foo@3HA
454 ?foo@3HA
455 ?foo@3HA
456 ?foo@3HA
457 ?foo@3HA
458 ?foo@3HA
459 ?foo@3HA
460 ?foo@3HA
461 ?foo@3HA
462 ?foo@3HA
463 ?foo@3HA
464 ?foo@3HA
465 ?foo@3HA
466 ?foo@3HA
467 ?foo@3HA
468 ?foo@3HA
469 ?foo@3HA
470 ?foo@3HA
471 ?foo@3HA
472 ?foo@3HA
473 ?foo@3HA
474 ?foo@3HA
475 ?foo@3HA
476 ?foo@3HA
477 ?foo@3HA
478 ?foo@3HA
479 ?foo@3HA
480 ?foo@3HA
481 ?foo@3HA
482 ?foo@3HA
483 ?foo@3HA
484 ?foo@3HA
485 ?foo@3HA
486 ?foo@3HA
487 ?foo@3HA
488 ?foo@3HA
489 ?foo@3HA
490 ?foo@3HA
491 ?foo@3HA
492 ?foo@3HA
493 ?foo@3HA
494 ?foo@3HA
495 ?foo@3HA
496 ?foo@3HA
497 ?foo@3HA
498 ?foo@3HA
499 ?foo@3HA
500 ?foo@3HA
501 ?foo@3HA
502 ?foo@3HA
503 ?foo@3HA
504 ?foo@3HA
505 ?foo@3HA
506 ?foo@3HA
507 ?foo@3HA
508 ?foo@3HA
509 ?foo@3HA
510 ?foo@3HA
511 ?foo@3HA
512 ?foo@3HA
513 ?foo@3HA
514 ?foo@3HA
515 ?foo@3HA
516 ?foo@3HA
517 ?foo@3HA
518 ?foo@3HA
519 ?foo@3HA
520 ?foo@3HA
521 ?foo@3HA
522 ?foo@3HA
523 ?foo@3HA
524 ?foo@3HA
525 ?foo@3HA
526 ?foo@3HA
527 ?foo@3HA
528 ?foo@3HA
529 ?foo@3HA
530 ?foo@3HA
531 ?foo@3HA
532 ?foo@3HA
533 ?foo@3HA
534 ?foo@3HA
535 ?foo@3HA
536 ?foo@3HA
537 ?foo@3HA
538 ?foo@3HA
539 ?foo@3HA
540 ?foo@3HA
541 ?foo@3HA
542 ?foo@3HA
543 ?foo@3HA
544 ?foo@3HA
545 ?foo@3HA
546 ?foo@3HA
547 ?foo@3HA
548 ?foo@3HA
549 ?foo@3HA
550 ?foo@3HA
551 ?foo@3HA
552 ?foo@3HA
553 ?foo@3HA
554 ?foo@3HA
555 ?foo@3HA
556 ?foo@3HA
557 ?foo@3HA
558 ?foo@3HA
559 ?foo@3HA
560 ?foo@3HA
561 ?foo@3HA
562 ?foo@3HA
563 ?foo@3HA
564 ?foo@3HA
565 ?foo@3HA
566 ?foo@3HA
567 ?foo@3HA
568 ?foo@3HA
569 ?foo@3HA
570 ?foo@3HA
571 ?foo@3HA
572 ?foo@3HA
573 ?foo@3HA
574 ?foo@3HA
575 ?foo@3HA
576 ?foo@3HA
577 ?foo@3HA
578 ?foo@3HA
579 ?foo@3HA
580 ?foo@3HA
581 ?foo@3HA
582 ?foo@3HA
583 ?foo@3HA
584 ?foo@3HA
585 ?foo@3HA
586 ?foo@3HA
587 ?foo@3HA
588 ?foo@3HA
589 ?foo@3HA
590 ?foo@3HA
591 ?foo@3HA
592 ?foo@3HA
593 ?foo@3HA
594 ?foo@3HA
595 ?foo@3HA
596 ?foo@3HA
597 ?foo@3HA
598 ?foo@3HA
599 ?foo@3HA
600 ?foo@3HA
601 ?foo@3HA
602 ?foo@3HA
603 ?foo@3HA
604 ?foo@3HA
605 ?foo@3HA
606 ?foo@3HA
607 ?foo@3HA
608 ?foo@3HA
609 ?foo@3HA
610 ?foo@3HA
611 ?foo@3HA
612 ?foo@3HA
613 ?foo@3HA
614 ?foo@3HA
615 ?foo@3HA
616 ?foo@3HA
617 ?foo@3HA
618 ?foo@3HA
619 ?foo@3HA
620 ?foo@3HA
621 ?foo@3HA
622 ?foo@3HA
623 ?foo@3HA
624 ?foo@3HA
625 ?foo@3HA
626 ?foo@3HA
627 ?foo@3HA
628 ?foo@3HA
629 ?foo@3HA
630 ?foo@3HA
631 ?foo@3HA
632 ?foo@3HA
633 ?foo@3HA
634 ?foo@3HA
635 ?foo@3HA
636 ?foo@3HA
637 ?foo@3HA
638 ?foo@3HA
639 ?foo@3HA
640 ?foo@3HA
641 ?foo@3HA
642 ?foo@3HA
643 ?foo@3HA
644 ?foo@3HA
645 ?foo@3HA
646 ?foo@3HA
647 ?foo@3HA
648 ?foo@3HA
649 ?foo@3HA
650 ?foo@3HA
651 ?foo@3HA
652 ?foo@3HA
653 ?foo@3HA
654 ?foo@3HA
655 ?foo@3HA
656 ?foo@3HA
657 ?foo@3HA
658 ?foo@3HA
659 ?foo@3HA
660 ?foo@3HA
661 ?foo@3HA
662 ?foo@3HA
663 ?foo@3HA
664 ?foo@3HA
665 ?foo@3HA
666 ?foo@3HA
667 ?foo@3HA
668 ?foo@3HA
669 ?foo@3HA
670 ?foo@3HA
671 ?foo@3HA
672 ?foo@3HA
673 ?foo@3HA
674 ?foo@3HA
675 ?foo@3HA
676 ?foo@3HA
677 ?foo@3HA
678 ?foo@3HA
679 ?foo@3HA
680 ?foo@3HA
681 ?foo@3HA
682 ?foo@3HA
683 ?foo@3HA
684 ?foo@3HA
685 ?foo@3HA
686 ?foo@3HA
687 ?foo@3HA
688 ?foo@3HA
689 ?foo@3HA
690 ?foo@3HA
691 ?foo@3HA
692 ?foo@3HA
693 ?foo@3HA
694 ?foo@3HA
695 ?foo@3HA
696 ?foo@3HA
697 ?foo@3HA
698 ?foo@3HA
699 ?foo@3HA
700 ?foo@3HA
701 ?foo@3HA
702 ?foo@3HA
703 ?foo@3HA
704 ?foo@3HA
705 ?foo@3HA
706 ?foo@3HA
707 ?foo@3HA
708 ?foo@3HA
709 ?foo@3HA
710 ?foo@3HA
711 ?foo@3HA
712 ?foo@3HA
713 ?foo@3HA
714 ?foo@3HA
715 ?foo@3HA
716 ?foo@3HA
717 ?foo@3HA
718 ?foo@3HA
719 ?foo@3HA
720 ?foo@3HA
721 ?foo@3HA
722 ?foo@3HA
723 ?foo@3HA
724 ?foo@3HA
725 ?foo@3HA
726 ?foo@3HA
727 ?foo@3HA
728 ?foo@3HA
729 ?foo@3HA
730 ?foo@3HA
731 ?foo@3HA
732 ?foo@3HA
733 ?foo@3HA
734 ?foo@3HA
735 ?foo@3HA
736 ?foo@3HA
737 ?foo@3HA
738 ?foo@3HA
739 ?foo@3HA
740 ?foo@3HA
741 ?foo@3HA
742 ?foo@3HA
743 ?foo@3HA
744 ?foo@3HA
745 ?foo@3HA
746 ?foo@3HA
747 ?foo@3HA
748 ?foo@3HA
749 ?foo@3HA
750 ?foo@3HA
751 ?foo@3HA
752 ?foo@3HA
753 ?foo@3HA
754 ?foo@3HA
755 ?foo@3HA
756 ?foo@3HA
757 ?foo@3HA
758 ?foo@3HA
759 ?foo@3HA
760 ?foo@3HA
761 ?foo@3HA
762 ?foo@3HA
763 ?foo@3HA
764 ?foo@3HA
765 ?foo@3HA
766 ?foo@3HA
767 ?foo@3HA
768 ?foo@3HA
769 ?foo@3HA
770 ?foo@3HA
771 ?foo@3HA
772 ?foo@3HA
773 ?foo@3HA
774 ?foo@3HA
775 ?foo@3HA
776 ?foo@3HA
777 ?foo@3HA
778 ?foo@3HA
779 ?foo@3HA
780 ?foo@3HA
781 ?foo@3HA
782 ?foo@3HA
783 ?foo@3HA
784 ?foo@3HA
785 ?foo@3HA
786 ?foo@3HA
787 ?foo@3HA
788 ?foo@3HA
789 ?foo@3HA
790 ?foo@3HA
791 ?foo@3HA
792 ?foo@3HA
793 ?foo@3HA
794 ?foo@3HA
795 ?foo@3HA
796 ?foo@3HA
797 ?foo@3HA
798 ?foo@3HA
799 ?foo@3HA
800 ?foo@3HA
801 ?foo@3HA
802 ?foo@3HA
803 ?foo@3HA
804 ?foo@3HA
805 ?foo@3HA
806 ?foo@3HA
807 ?foo@3HA
808 ?foo@3HA
809 ?foo@3HA
810 ?foo@3HA
811 ?foo@3HA
812 ?foo@3HA
813 ?foo@3HA
814 ?foo@3HA
815 ?foo@3HA
816 ?foo@3HA
817 ?foo@3HA
818 ?foo@3HA
819 ?foo@3HA
820 ?foo@3HA
821 ?foo@3HA
822 ?foo@3HA
823 ?foo@3HA
824 ?foo@3HA
825 ?foo@3HA
826 ?foo@3HA
827 ?foo@3HA
828 ?foo@3HA
829 ?foo@3HA
830 ?foo@3HA
831 ?foo@3HA
832 ?foo@3HA
833 ?foo@3HA
834 ?foo@3HA
835 ?foo@3HA
836 ?foo@3HA
837 ?foo@3HA
838 ?foo@3HA
839 ?foo@3HA
840 ?foo@3HA
841 ?foo@3HA
842 ?foo@3HA
843 ?foo@3HA
844 ?foo@3HA
845 ?foo@3HA
846 ?foo@3HA
847 ?foo@3HA
848 ?foo@3HA
849 ?foo@3HA
850 ?foo@3HA
851 ?foo@3HA
852 ?foo@3HA
853 ?foo@3HA
854 ?foo@3HA
855 ?foo@3HA
856 ?foo@3HA
857 ?foo@3HA
858 ?foo@3HA
859 ?foo@3HA
860 ?foo@3HA
861 ?foo@3HA
862 ?foo@3HA
863 ?foo@3HA
864 ?foo@3HA
865 ?foo@3HA
866 ?foo@3HA
867 ?foo@3HA
868 ?foo@3HA
869 ?foo@3HA
870 ?foo@3HA
871 ?foo@3HA
872 ?foo@3HA
873 ?foo@3HA
874 ?foo@3HA
875 ?foo@3HA
876 ?foo@3HA
877 ?foo@3HA
878 ?foo@3HA
879 ?foo@3HA
880 ?foo@3HA
881 ?foo@3HA
882 ?foo@3HA
883 ?foo@3HA
884 ?foo@3HA
885 ?foo@3HA
886 ?foo@3HA
887 ?foo@3HA
888 ?foo@3HA
889 ?foo@3HA
890 ?foo@3HA
891 ?foo@3HA
892 ?foo@3HA
893 ?foo@3HA
894 ?foo@3HA
895 ?foo@3HA
896 ?foo@3HA
897 ?foo@3HA
898 ?foo@3HA
899 ?foo@3HA
900 ?foo@3HA
901 ?foo@3HA
902 ?foo@3HA
903 ?foo@3HA
904 ?foo@3HA
905 ?foo@3HA
906 ?foo@3HA
907 ?foo@3HA
908 ?foo@3HA
909 ?foo@3HA
910 ?foo@3HA
911 ?foo@3HA
912 ?foo@3HA
913 ?foo@3HA
914 ?foo@3HA
915 ?foo@3HA
916 ?foo@3HA
917 ?foo@3HA
918 ?foo@3HA
919 ?foo@3HA
920 ?foo@3HA
921 ?foo@3HA
922 ?foo@3HA
923 ?foo@3HA
924 ?foo@3HA
925 ?foo@3HA
926 ?foo@3HA
927 ?foo@3HA
928 ?foo@3HA
929 ?foo@3HA
930 ?foo@3HA
931 ?foo@3HA
932 ?foo@3HA
933 ?foo@3HA
934 ?foo@3HA
935 ?foo@3HA
936 ?foo@3HA
937 ?foo@3HA
938 ?foo@3HA
939 ?foo@3HA
940 ?foo@3HA
941 ?foo@3HA
942 ?foo@3HA
943 ?foo@3HA
944 ?foo@3HA
945 ?foo@3HA
946 ?foo@3HA
947 ?foo@3HA
948 ?foo@3HA
949 ?foo@3HA
950 ?foo@3HA
951 ?foo@3HA
952 ?foo@3HA
953 ?foo@3HA
954 ?foo@3HA
955 ?foo@3HA
956 ?foo@3HA
957 ?foo@3HA
958 ?foo@3HA
959 ?foo@3HA
960 ?foo@3HA
961 ?foo@3HA
962 ?foo@3HA
963 ?foo@3HA
964 ?foo@3HA
965 ?foo@3HA
966 ?foo@3HA
967 ?foo@3HA
968 ?foo@3HA
969 ?foo@3HA
970 ?foo@3HA
971 ?foo@3HA
972 ?foo@3HA
973 ?foo@3HA
974 ?foo@3HA
975 ?foo@3HA
976 ?foo@3HA
977 ?foo@3HA
978 ?foo@3HA
979 ?foo@3HA
980 ?foo@3HA
981 ?foo@3HA
982 ?foo@3HA
983 ?foo@3HA
984 ?foo@3HA
985 ?foo@3HA
986 ?foo@3HA
987 ?foo@3HA
988 ?foo@3HA
989 ?foo@3HA
990 ?foo@3HA
991 ?foo@3HA
992 ?foo@3HA
993 ?foo@3HA
994 ?foo@3HA
995 ?foo@3HA
996 ?foo@3HA
997 ?foo@3HA
998 ?foo@3HA
999 ?foo@3HA
1000 ?foo@3HA
1001 ?foo@3HA
1002 ?foo@3HA
1003 ?foo@3HA
1004 ?foo@3HA
1005 ?foo@3HA
1006 ?foo@3HA
1007 ?foo@3HA
1008 ?foo@3HA
1009 ?foo@3HA
1010 ?foo@3HA
1011 ?foo@3HA
1012 ?foo@3HA
1013 ?foo@3HA
1014 ?foo@3HA
1015 ?foo@3HA
1016 ?foo@3HA
1017 ?foo@3HA
1018 ?foo@3HA
1019 ?foo@3HA
1020 ?foo@3HA
1021 ?foo@3HA
1022 ?foo@3HA
1023 ?foo@3HA
1024 ?foo@3HA
1025 ?foo@3HA
1026 ?foo@3HA
1027 ?foo@3HA
1028 ?foo@3HA
1029 ?foo@3HA
1030 ?foo@3HA
1031 ?foo@3HA
1032 ?foo@3HA
1033 ?foo@3HA
1034 ?foo@3HA
1035 ?foo@3HA
1036 ?foo@3HA
1037 ?foo@3HA
1038 ?foo@3HA
1039 ?foo@3HA
1040 ?foo@3HA
1041 ?foo@3HA
1042 ?foo@3HA
1043 ?foo@3HA
1044 ?foo@3HA
1045 ?foo@3HA
1046 ?foo@3HA
1047 ?foo@3HA
1048 ?foo@3HA
1049 ?foo@3HA
1050 ?foo@3HA
1051 ?foo@3HA
1052 ?foo@3HA
1053 ?foo@3HA
1054 ?foo@3HA
1055 ?foo@3HA
1056 ?foo@3HA
1057 ?foo@3HA
1058 ?foo@3HA
1059 ?foo@3HA
1060 ?foo@3HA
1061 ?foo@3HA
1062 ?foo@3HA
1063 ?foo@3HA
1064 ?foo@3HA
1065 ?foo@3HA
1066 ?foo@3HA
1067 ?foo@3HA
1068 ?foo@3HA
1069 ?foo@3HA
1070 ?foo@3HA
1071 ?foo@3HA
1072 ?foo@3HA
1073 ?foo@3HA
1074 ?foo@3HA
1075 ?foo@3HA
1076 ?foo@3HA
1077 ?foo@3HA
1078 ?foo@3HA
1079 ?foo@3HA
1080 ?foo@3HA
1081 ?foo@3HA
1082 ?foo@3HA
1083 ?foo@3HA
1084 ?foo@3HA
1085 ?foo@3HA
1086 ?foo@3HA
1087 ?foo@3HA
1088 ?foo@3HA
1089 ?foo@3HA
1090 ?foo@3HA
1091 ?foo@3HA
1092 ?foo@
```

- 以MS-WIN下的应用程序为例，从高地址到低地址，自上而下的是：
  - 静态数据区
    - 全局和静态量表
  - 代码区
  - 运行栈
  - 内存堆

Diagram illustrating memory layout from high address to low address (top to bottom):

- 全局和静态量表 (Global and Static Data)
- 代码区 (Code)
- 运行栈 (Stack)
- 内存堆 (Heap)

地址：高


↓

↑


地址：低

## 12.2.2 程序运行栈的设计

- **子程序/函数**运行时所需的基本空间：活动记录
- **进入**子程序/函数时分配，地址空间向下生长（从高地址到低地址）
- 从子程序/函数**返回时**，当前运行栈将被废弃
- 递归调用的同一个子程序函数，每次调用都将获得**独立**的运行栈空间，从而保证了递归程序和多线程程序的正确运行。



Beihang University



```

1 // C12P1.cpp
2 #include "stdafx.h"
3 int global_val = 0;
4
5 int foo(int n) {
6     static static_val = 0;
7     int i;
8     for(i=0; i<100; i++) {
9         n = n*i + global_val + static_val;
10    }
11    static_val = n / 2;
12    printf("static_val is %x\n",
13           static_val);
14    return n;
15 }
16
17 main(int argc, char* argv[]) {
18     global_val = 99;
19     int n = foo(100);
20     printf("foo(100) is %x\n",n);
21     return 0;
22 }

```

1 PUBLIC	%foo@%YAHII@Z.foo
2 PUBLIC	?_C__0B_C12P1@%static_val%75cfd7w%SAA@; string
3 EXTEN	PRNT_NLAK
4 DATA	SEGMENT
5 ?_C__0B_C12P1@%static_val%75cfd7w%SAA@ DB 'static_val is %x', 0Ah, 0Dh, ; string	
6 .DATA	ENDS
7 .TEXT	SEGMENT
8 _dch-8-	
9 %foo@%YAHII@Z PROC NEAR ; foo	
10 mov	ecx, DWORD PTR [global_val+3SHA; global_val
11 mov	edx, DWORD PTR '_static_val'+171760@%YAHII@Z+4EHA,
12 push	edi
13 mov	edi, DWORD PTR _rcx[esp]
14 push	edi
15 xor	ecx, ecx
16 LSL3:	
17 mov	edi, ecx
18 imul	edi, esi
19 add	esi, edi
20 add	edi, ecx
21 cmp	ecx, 100
22 jge	JMP SHORT LSL3
23 mov	esi, edi
24 j	JMP SHORT LSL3
25 mov	esi, esi
26 mov	ecx, esi
27 mov	ecx, esi
28 sub	esi, 1
29 jmp	short ecx
30 _C__0B_C12P1@%p_u_s_b O F S E S t	
31 FLAT?_C__0B_C12P1@%p_u_s_b O F S E S t ; string	
32 mov	DWORD PTR '_static_val'+171760@%YAHII@Z+4EHA,
33 call	_printf
34 jmp	exp.8
35 mov	edi, esi
36 call	exp.8
37 pop	edi
38 %foo@%YAHII@Z ENDP ; foo	
39 .TEXT	ENDS

北京航空航天大学计算机学院

# 函数foo的运行栈示例

The diagram illustrates the stack frame for a function named 'foo'. It consists of a vertical stack of six boxes. From top to bottom, the boxes are labeled: 'n', 'ret\_addr', 'ESI', 'EDI', 'EAX', and 'String ADDR'. To the left of the stack, two pointers are shown: 'ESP0' with an arrow pointing to the 'ESI' box, and 'ESP2' with an arrow pointing to the 'EAX' box. To the right of the stack, two pointers are shown: 'ESP1' with an arrow pointing to the 'EDI' box, and 'ESP3' with an arrow pointing to the 'String ADDR' box. The 'ESI' and 'EDI' boxes are connected by a horizontal line, as are the 'EAX' and 'String ADDR' boxes.

北京航空航天大学计算机学院

Beihang University logo with the text 'Beihang University' and 'Excellence in BUSA SET'.

- 一个典型的运行栈包括（活动记录）
  - 函数的返回地址
  - 全局寄存器的保存区
  - 临时变量的保存区
  - 未分配到全局寄存器的局部变量的保存区
  - 其他辅助信息的保存区
    - 例，PASCAL/PL-I类语言的DISPLAY区

Diagram illustrating the RISC architecture stack frame layout (from high address to low address):

- arg<sub>n</sub>, ..., arg<sub>1</sub>: Parameters (greater than 4 parameters)
- lr: Caller's return address
- r4: Global register save area
- Temp var: Temporary variable save area
- local var: Unallocated local variables
- sp: Stack pointer, save area

## 12.4 寄存器的分配和指派

- 为什么要分配和管理寄存器？
  - 寄存器的访问速度是所有存储形式中最快的，
  - 某些运算只能发生在寄存器当中
  - 从程序优化的角度来说，我们希望所有指令的执行都仅在寄存器中完成
  - 而资源是有限的

北京航空航天大学计算机学院

- 寄存器通常分为
  - 通用寄存器**
    - X86: EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP, etc
    - ARM: R0-R15, etc
  - 专用寄存器**
    - X86: 浮点寄存器栈, etc
- 通用寄存器
  - 保留寄存器
    - 例如，X86的ESP栈指针寄存器，ARM的返回寄存器LR
  - 调用方保存的寄存器——**临时寄存器**
    - caller-saved register
    - X86: EAX, ECX, EDX
    - ARM: R0-R3, R12, LR
  - 被调用方保存的寄存器——**全局寄存器**
    - callee-saved register
    - X86: EBX, ESI, EDI, EBP
    - ARM: R4-R11

北京航空航天大学计算机学院

### 12.4.1 全局寄存器分配

- 全局寄存器分配：**
  - “**全局**”相对于“基本块”而言，不是“程序全局”
- 分配原则**
  - 优先分配给跨基本块仍然活跃的变量，尤其是循环体内最活跃的变量
  - 局部变量参与全局寄存器分配
    - 为了线程安全，全局变量静态变量一般不参与全局寄存器分配。

**寄存器专属于线程！**

北京航空航天大学计算机学院

### 如果全局/静态量参与寄存器分配？

```

Thread 1:
void foo(int a)
{
    static int s_c = 0;
    a = 1;
    s_c = 1;
    s_c += a;
}

Thread 2:
void foo(int a)
{
    static int s_c = 0;
    a = 2;
    s_c = 1;
    s_c += a;
}

Context switch!

Thread 1:
void foo(int a)
{
    static int s_c = 0;
    a = 1;
    s_c (ESI) = 1;
    s_c += a;
}

Thread 2:
void foo(int a)
{
    static int s_c = 0;
    a = 2;
    s_c (ESI) = 0;
    s_c += a;
}

Context switch!

Thread 1:
void foo(int a)
{
    static int s_c = 0;
    a = 1;
    s_c (ESI) = 1;
    s_c += a;
}

Thread 2:
void foo(int a)
{
    static int s_c = 0;
    a = 2;
    s_c (ESI) = 2;
    s_c += a;
}

```

北京航空航天大学计算机学院

### 常用全局寄存器分配方法

- 引用计数**
  - 通过统计变量在函数内被引用的次数，并根据被引用的特点赋予不同的权重，最终为每个变量计算出一个唯一的权值，根据权值的大小排序，将全局寄存器依次分配给权值最大的变量
- 着色图算法**
  - 通过构建变量之间的**冲突图**，在图上应用着色算法，将不同的全局寄存器分配给有冲突的变量。

北京航空航天大学计算机学院

#### 12.4.1.1 引用计数

- 原则：**如果一个局部变量被访问的次数较多，那么它获得全局寄存器的机会也较大
- 注意：**出现在循环，尤其是内层嵌套循环中的变量的被访问次数应该得到一定的加权

3个局部变量，2个全局寄存器可供分配，谁将获得寄存器？

```

B1
j := b + 1
B2
a := a + 1
j := j * a
if j < 1000 goto B2
B3
b := b + 1
j := b + 2

```

j 5次  
b 4次  
a 3次

北京航空航天大学计算机学院

## 引用计数

- **分配算法**: 如果有N个全局寄存器可供分配, 则前N个变量拥有全局寄存器, 其余变量在程序运行栈(活动记录)分配存储单元
- **问题**: 不再使用的变量不能及时释放寄存器
  - 如变量a在前期大量使用, 后端程序中不适用了
- **解决办法**:
  - 活跃变量分析、冲突图
  - 着色算法

## 12.4.1.2 图着色算法

- 一种简化的图着色算法
  - 步骤:
    - 1、通过数据流分析, 构建变量的冲突图
    - 2、如果可供分配k个全局寄存器, 那么我们就尝试用k种颜色给该冲突图着色

### • 步骤1、通过数据流分析, 构建变量的冲突图

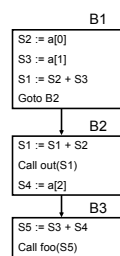
– 什么是变量的冲突图?

- 它的结点是待分配全局寄存器的变量
- 当两个变量中的一个变量在另一个变量定义(赋值)处是活跃的, 它们之间便有一条边连接 所谓变量在代码n处活跃, 是指程序运行时变量在n处拥有的值, 在从n出发的某条路径上会被使用。
- 直观的理解, 可以认为**有边相连的变量, 它们无法共用一个全局寄存器, 或者同一存储单元, 否则程序运行将可能出错**
- **无连接关系的变量, 即便它们占用同一全局寄存器, 或同一存储单元, 程序运行也不会出错**

### • 例:

```
for 每个基本块B do in[B] = ∅;
while 尚有变量变化 do
    for 每个基本块B do begin
        out[B] = U ∪ use[B], in[B]
        in[B] = use[B] ∪ out[B] - def[B]
    end
end
```

流图



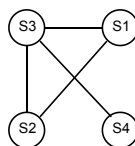
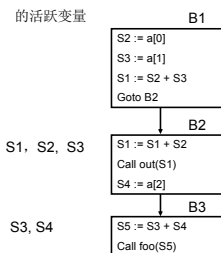
def[B]	use[B]	in[B]	out[B]
S1,S2,S3	∅	∅	S1,S2,S3
S4	S1,S2	S1,S2,S3	S3,S4
S5	S3,S4	S3,S4	∅
		∅	

### • 例:

基本块入口处的活跃变量

流图

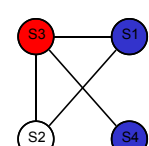
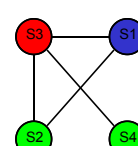
冲突图



### • 步骤2、如果可供分配k个全局寄存器, 那么我们就尝试用k种颜色给该冲突图着色

假设1: k=3, R0,R1,R2

假设2: k=2, R0,R1



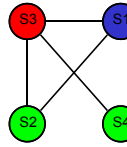
```

S2 := a[0]
S3 := a[1]
S1 := S2 + S3
Goto B2

S1 := S1 + S2
Call out(S1)
S4 := a[2]

S5 := S3 + S4
Call foo(S5)
    
```

假设1:  $k=3$ ,  $R0, R1, R2$



```

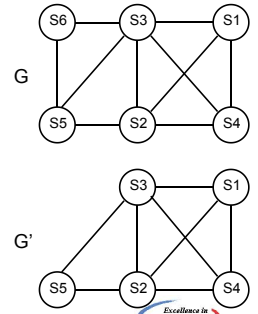
R2 := a[0]
R0 := a[1]
R1 := R2 + R0
Goto B2

R1 := R1 + R2
Call out(R1)
R2 := a[2]

S5 := R0 + R2
Call foo(S5)
    
```

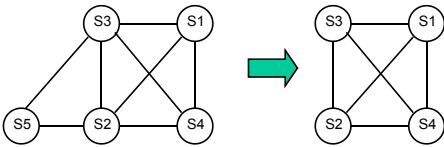
## 算法12.2 一种启发式图着色算法

- 冲突图G
  - 寄存器数目为K
  - 假设  $K=3$

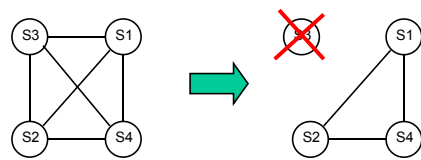


- 步骤1、找到第一个连接边数目小于K的结点，将它从图G中移走，形成图G'

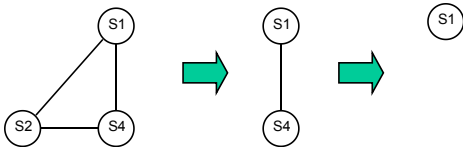
- 步骤2、重复步骤1，直到无法再从G'中移走结点



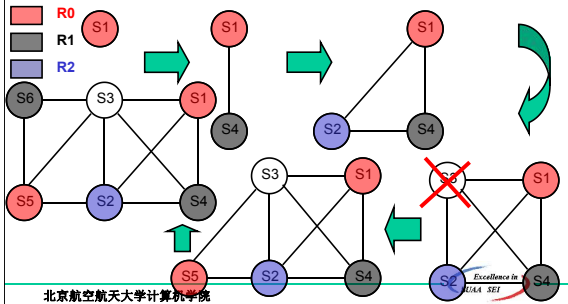
- 步骤3、在图中选取适当的结点，将它记录为“不分配全局寄存器”的结点，并从图中移走



- 步骤4、重复步骤1~步骤3，直到图中仅剩余1个结点



- 步骤5、给剩余的最后一个结点选取一种颜色，然后按照结点被移走的顺序，反向将结点和边添加进去，并依次给新加入的结点选取颜色。（保证有链接边的结点着不同的颜色）





## 12.4.2 临时寄存器分配

- 为什么在代码生成过程中，需要对临时寄存器进行管理？
  - 因为生成某些指令时，必须使用指定寄存器
  - 临时寄存器中保存有此前的计算中间结果
- 以X86为例，生成代码时可用的临时寄存器
  - EAX, ECX, EDX等

## 临时寄存器的管理原则和方法

- 临时寄存器的生存范围
  - 不超越基本块
  - 不跨越函数调用
- 临时寄存器的管理方法
  - 寄存器池

### 临时寄存器池的使用方法：设EAX, EDX可用

全局寄存器分配结果：

a	EBX
b	ESI
c	EDI

临时变量在运行栈上的保存地址：

t3	ESP+10H
t2	ESP+0CH
t1	ESP+08H

寄存器池：

t1 := -c

t1	EAX
	EDX

mov EAX, EDI  
neg EAX

t2 := t1 - b

t1	EAX
t2	EDX

mov EDX, EAX  
sub EDX, ESI

t3 := t2 + t2

t3	EAX
t2	EDX

mov [ESP+08H], EAX  
mov EAX, EDX  
add EAX, EAX

a := t3

t3	EAX
t2	EDX

mov EBX, EAX

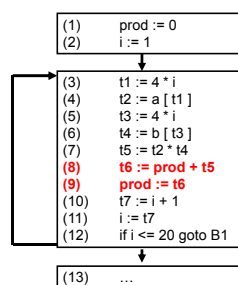
### 临时寄存器的分配和管理方法

- 进入基本块时，清空临时寄存器池
- 为当前中间代码生成目标代码时，无论临时变量还是局部变量（亦或全局变量和静态变量），如需使用临时寄存器，都可以向临时寄存器池申请
  - 如寄存器池中有空闲寄存器，则可将该寄存器标识为被该申请变量占用，并返回该空闲寄存器
  - 如寄存器池中无空闲寄存器，则选取一个在即将生成代码中不会被使用的寄存器写回相应的内存空间，标识该寄存器被新的变量占用，返回该寄存器
- 在基本块结尾，或者函数调用发生前，将寄存器池中所有被占用的临时寄存器写回相应的内存空间，清空临时寄存器池

## 12.5 指令选择

- 不同的体系结构采用了不同类型的指令集，由于体系结构和指令集的差异，使得在生成代码时需要采用不同的指令选择策略
  - RISC
    - ARM, MIPS
  - CISC
    - X86
  - VLW/EPIC
    - Itanium

### 例：



### • RISC: ARM

– prod = R5  
– t5 = [SP+8]  
– t6 = R2

ldr R3, [SP, #8] ; R3 = t5

add R5, R2, R3 ; prod = t6 + R3

### • CISC: X86

– prod = EBX  
– t5 = [ESP+8]  
– t6 = ECX

mov ECX, EBX ; t6 = prod  
add ECX, [ESP+8] ; t6 = prod + t5  
mov EBX, ECX ; prod = t6

Compiler

(1) t1 = -c

例: (2) t2 = t1 - b

(3) t3 = t2 + t2

(4) a = t3

逐句转换:

(1) mov ECX, EDI

neg ECX

mov [ESP+08H], ECX

(2) mov ECX, [ESP+08H]

sub ECX, ESI

mov [ESP+0CH], ECX

(3) mov ECX, [ESP+0CH]

add ECX, [ESP+0CH]

mov [ESP+10H], ECX

(4) mov EBX, [ESP+10H]

逐句转换+临时寄存器池:

(1) mov EAX, EDI

neg EAX

(2) mov EDX, EAX

sub EDX, ESI

(3) mov [ESP+08H], EAX

mov EAX, EDX

add EAX, EAX

(4) mov EBX, EAX

a	EBX	t3	ESP+10H
b	ESI	t2	ESP+0CH
c	EDI	t1	ESP+08H

北京航空航天大学计算机学院

Compiler

作业:

新编教材第十五章1,4,5,6

北京航空航天大学计算机学院

Compiler

优化部分小结

第11章:代码优化

与机器无关的优化独立于机器的(中间)代码优化

与机器有关的优化目标代码上的优化(与具体机器有关)

优化分为两大类

北京航空航天大学计算机学院

Compiler

优化方法的分类2:

- 局部优化技术
  - 指在基本块内进行的优化
  - 例如,局部公共子表达式删除
- 全局优化技术
  - 函数/过程内进行的优化
  - 跨越基本块
  - 例如,全局数据流分析
- 跨函数优化技术
  - 整个程序
  - 例如,跨函数别名分析,逃逸分析等

北京航空航天大学计算机学院

Compiler

代码优化

- DAG图
  - 消除局部公共子表达式
  - 从DAG图导出中间代码的启发式算法
- 数据流分析
  - 到达定义分析
  - 活跃变量分析
- 构建冲突图
  - 变量冲突的基本概念
  - 通过活跃变量分析构建(精度不太高的)冲突图

北京航空航天大学计算机学院

Compiler

十二章 代码生成

- 微处理器体系结构基础知识
- 全局寄存器分配算法
  - 引用计数
  - 图着色
- 临时寄存器池管理方法与指令选择

北京航空航天大学计算机学院