

Sistemas de Computação

STACK FRAME

- 1º Guardar o valor dos registos do caller nome (normalmente não aparece)
- 2º Guardar os argumentos da função.
- 3º Guardar o endereço de retorno para a main (próxima instrução)
- 4º Push do base pointer → guarda o base pointer
- 5º Copiar o valor do base pointer para o %esp.
- 6º Guardar os conteúdos dos registos do callee nome
- 7º Guardar, se necessário, o valor de variáveis locais (caso o código é otimizado, as variáveis ficam guardadas em registos)

%esp → over local calle nome antigo %ebp endereço currents ↓

Registers:

Prog. em C (text)	↓ gcc → compilador	%eax → valores e endereços
Prog. Assembly (text)	↓ gcc → (Assembly)	%ecr → calle nome
Prog. object (binário)	↓ gcc (linker)	%edc → (funções chamadas)
Prog. executável (binário)	↓ %esp → stack pointer	%edi → calle nome
		%esi → (variáveis)
		%ebp → base / frame pointer

G1.00: faz mto acima à memória, quando se guardam as variáveis

Instâncias de callt0

o offset é sempre calculado um endereço ao endereço seguinte!

Diretiva = End. seguinte + Tamanho da Instrução

G1.02: novas variáveis não geridas das estruturas, o que diminui o espaço à memória.

Exemplos:

- 1º 0x700C5C0 E1 28 ; jmp 0x700C5EA  
0x700C5C2 ← 0x700C5C1 + 28
- 2º 0x700C5C0 E2 FDEF ; jmp 0x700C5C0  
0x700C5C3 ← 0x700C5C0 + FFFF
- 3º 0x700C5C0 E1 (xx) ; jmp 0x700C5E0  
0x700C5C2 ← 0x700C5E0 - 0x700C5C2  
= 0x000001E

Pseudodemolação na memória:

mar = 0x01234567; &mar = 0x100

0x100	0x101
67	45 23 01

RISC versus IA-32

RISC: conjunto reduzido de instruções

IA-32: 32 operações IA-32: 2 operações

Técnicas de PCE/PRC

1º Explicação de instruções

a) end. base de a → mod(%edx,%eax)  
a[4] end. base + resto (a[4]) → mod(%eax,%edx,4)/%eax  
a[5:7] corrigimento do end. base (mod(%eax,%edx,4)%edx)

2º Para verificar a paridade de cada elemento a[i]:

- Se a[i] for par vai dar zero; - Se a[i] for ímpar vai dar 1 (8 → And bit a bit) Exemplo: 128 | 128 + 1101
- Em -02 a instrução é o tñtil \$1, %eax que 128 + 1101 não guarda o resultado apenas altera os flags, mantendo a zero flag

3º mod(%eax,%edx,4)%eax

Guarda o valor de a[i] em registo (a[i]:=1)

imod %eax

Guarda um valor de a[i]

mod %eax (%eax,%edx,4)

Guarda o valor de a[i]+1 mod memória

5º a) Verificando o valor do esp = 0x8048364 podemos afirmar que a próxima instrução é mov(%eax,%edx,4)%eax

b) já foram executadas 2 instruções escritas 20.

c) 0x8049620

Exemplo:

```
int norma_guardar(int m, int *a) {
    int norma = 0, i;
    for (i = 0; i < m; i++) {
        if (a[i] > 100) norma += a[i];
    }
    return norma;
}
```

Vale.	Reg.	esp
norma	%eax	calle nome
a[i]	%eax	%esp
i	%eax	antigo %ebp
m	%eax	end. Reg.

parâmetros

Diretivas - 1 byte  
alinhado - 2 "  
int - double word - 4  
longint - double word - 8  
float - single precision - 4  
double - double precision - 8  
longdouble - extended precision - 10/12

frame pointer - double word - 4

Acesso a Arreglos:

ext. mem.: a[tam] [tamanho do array + índice] tamanhos

assembly : (%edx, %eax, 4) → array de inteiros

Exemplo: int a[5];

# edx = %d; # maior (%edx, %eax, 4), %eax => a[1];

# eax = %d; # menor (%edx, %eax, 4), %eax => 8 a[1];

Estruturas de Controlo:

if - then - else

if (cond) goto true;

else goto false;

true then - statement;

false then - statement;

else;

do while

loop:

if (!cond) goto done; body statement

body statement

goto loop;

done;

while (cond)

if (cond) body statement

body statement

goto loop;

done;

for

loop - inicial;

while (cond) { body statement }

body statement

(cond, update);

done;

body - statement

6º a) Endereço de saída: 0x080483d4  
0x080483d4 + 5 = 0x080483d9

7º 0x80483d4 + ?? = 8048867  
?? = 0x80483d4 - 0x80483d7  
?? = 1d

8º 0x80483d4 + ?? = 8048867

9º 0x80483d4 - 0x80483d7

10º 0x80483d4 - 1d

11º 0x80483d4 - 1d

12º 0x80483d4 - 1d

13º 0x80483d4 - 1d

14º 0x80483d4 - 1d

15º 0x80483d4 - 1d

16º 0x80483d4 - 1d

17º 0x80483d4 - 1d

18º 0x80483d4 - 1d

19º 0x80483d4 - 1d

20º 0x80483d4 - 1d

21º 0x80483d4 - 1d

22º 0x80483d4 - 1d

23º 0x80483d4 - 1d

24º 0x80483d4 - 1d

25º 0x80483d4 - 1d

26º 0x80483d4 - 1d

27º 0x80483d4 - 1d

28º 0x80483d4 - 1d

29º 0x80483d4 - 1d

30º 0x80483d4 - 1d

31º 0x80483d4 - 1d

32º 0x80483d4 - 1d

33º 0x80483d4 - 1d

34º 0x80483d4 - 1d

35º 0x80483d4 - 1d

36º 0x80483d4 - 1d

37º 0x80483d4 - 1d

38º 0x80483d4 - 1d

39º 0x80483d4 - 1d

40º 0x80483d4 - 1d

41º 0x80483d4 - 1d

42º 0x80483d4 - 1d

43º 0x80483d4 - 1d

44º 0x80483d4 - 1d

45º 0x80483d4 - 1d

46º 0x80483d4 - 1d

47º 0x80483d4 - 1d

48º 0x80483d4 - 1d

49º 0x80483d4 - 1d

50º 0x80483d4 - 1d

51º 0x80483d4 - 1d

52º 0x80483d4 - 1d

53º 0x80483d4 - 1d

54º 0x80483d4 - 1d

55º 0x80483d4 - 1d

56º 0x80483d4 - 1d

57º 0x80483d4 - 1d

58º 0x80483d4 - 1d

59º 0x80483d4 - 1d

60º 0x80483d4 - 1d

61º 0x80483d4 - 1d

62º 0x80483d4 - 1d

63º 0x80483d4 - 1d

64º 0x80483d4 - 1d

65º 0x80483d4 - 1d

66º 0x80483d4 - 1d

67º 0x80483d4 - 1d

68º 0x80483d4 - 1d

69º 0x80483d4 - 1d

70º 0x80483d4 - 1d

71º 0x80483d4 - 1d

72º 0x80483d4 - 1d

73º 0x80483d4 - 1d

74º 0x80483d4 - 1d

75º 0x80483d4 - 1d

76º 0x80483d4 - 1d

77º 0x80483d4 - 1d

78º 0x80483d4 - 1d

79º 0x80483d4 - 1d

80º 0x80483d4 - 1d

81º 0x80483d4 - 1d

82º 0x80483d4 - 1d

83º 0x80483d4 - 1d

84º 0x80483d4 - 1d

85º 0x80483d4 - 1d

86º 0x80483d4 - 1d

87º 0x80483d4 - 1d

88º 0x80483d4 - 1d

89º 0x80483d4 - 1d

90º 0x80483d4 - 1d

91º 0x80483d4 - 1d

92º 0x80483d4 - 1d

93º 0x80483d4 - 1d

94º 0x80483d4 - 1d

95º 0x80483d4 - 1d

96º 0x80483d4 - 1d

97º 0x80483d4 - 1d

98º 0x80483d4 - 1d

99º 0x80483d4 - 1d

100º 0x80483d4 - 1d

101º 0x80483d4 - 1d

102º 0x80483d4 - 1d

103º 0x80483d4 - 1d

104º 0x80483d4 - 1d

105º 0x80483d4 - 1d

106º 0x80483d4 - 1d

107º 0x80483d4 - 1d

108º 0x80483d4 - 1d

109º 0x80483d4 - 1d

110º 0x80483d4 - 1d

111º 0x80483d4 - 1d

112º 0x80483d4 - 1d

113º 0x80483d4 - 1d

114º 0x80483d4 - 1d

115º 0x80483d4 - 1d

116º 0x80483d4 - 1d

117º 0x80483d4 - 1d

118º 0x80483d4 - 1d

119º 0x80483d4 - 1d

120º 0x80483d4 - 1d

121º 0x80483d4 - 1d

122º 0x80483d4 - 1d

123º 0x80483d4 - 1d

124º 0x80483d4 - 1d

125º 0x80483d4 - 1d

126º 0x80483d4 - 1d

127º 0x80483d4 - 1d

128º 0x80483d4 - 1d

129º 0x80483d4 - 1d

130º 0x80483d4 - 1d

131º 0x80483d4 - 1d

132º 0x80483d4 - 1d

133º 0x80483d4 - 1d

134º 0x80483d4 - 1d

135º 0x80483d4 - 1d

136º 0x80483d4 - 1d

137º 0x80483d4 - 1d

138º 0x80483d4 - 1d

139º 0x80483d4 - 1d

140º 0x80483d4 - 1d

141º 0x80483d4 - 1d

142º 0x80483d4 - 1d

143º 0x80483d4 - 1d

144º 0x80483d4 - 1d

145º 0x80483d4 - 1d

146º 0x80483d4 - 1d

147º 0x80483d4 - 1d

148º 0x80483d4 - 1d

149º 0x80483d4 - 1d

150º 0x80483d4 - 1d

151º 0x80483d4 - 1d

152º 0x80483d4 - 1d

153º 0x80483d4 - 1d

154º 0x80483d4 - 1d

155º 0x80483d4 - 1d

156º 0x80483d4 - 1d

157º 0x80483d4 - 1d

158º 0x80483d4 - 1d

159º 0x80483d4 - 1d

160º 0x80483d4 - 1d

161º 0x80483d4 - 1d

162º 0x80483d4 - 1d

163º 0x80483d4 - 1d

164º 0x80483d4 - 1d

165º 0x80483d4 - 1d

166º 0x80483d4 - 1d

167º 0x80483d4 - 1d

168º 0x80483d4 - 1d

169º 0x80483d4 - 1d

170º 0x80483d4 - 1d

171º 0x80483d4 - 1d

172º 0x80483d4 - 1d

173º 0x80483d4 - 1d

174º 0x80483d4 - 1d

175º 0x80483d4 - 1d

176º 0x80483d4 - 1d

177º 0x80483d4 - 1d

178º 0x80483d4 - 1d

179º 0x80483d4 - 1d

180º 0x80483d4 - 1d

181º 0x80483d4 - 1d

182º 0x80483d4 - 1d

183º 0x80483d4 - 1d

184º 0x80483d4 - 1d

185º 0x80483d4 - 1d

186º 0x80483d4 - 1d

187º 0x80483d4 - 1d

188º 0x80483d4 - 1d

189º 0x80483d4 - 1d

190º 0x80483d4 - 1d

191º 0x80483d4 - 1d

192º 0x80483d4 - 1d

193º 0x80483d4 - 1d

194º 0x80483d4 - 1d

195º 0x80483d4 - 1d

196º 0x80483d4 - 1d

197º 0x80483d4 - 1d

198º 0x80483d4 - 1d

199º 0x80483d4 - 1d

200º 0x80483d4 - 1d

201º 0x80483d4 - 1d

202º 0x80483d4 - 1d

203º 0x80483d4 - 1d

204º 0x80483d4 - 1d

205º 0x80483d4 - 1d

206º 0x80483d4 - 1d

207º 0x80483d4 - 1d

208º 0x80483d4 - 1d

209º 0x80483d4 - 1d

210º 0x80483d4 - 1d

211º 0x80483d4 - 1d

212º 0x80483d4 - 1d

213º 0x80483d4 - 1d

214º 0x80483d4 - 1d

215º 0x80483d4 - 1d

216º 0x80483d4 - 1d

217º 0x80483d4 - 1d

218º 0x80483d4 - 1d

219º 0x80483d4 - 1d

220º 0x80483d4 - 1d

221º 0x80483d4 - 1d

222º 0x80483d4 - 1d

223º 0x80483d4 - 1d

224º 0x80483d4 - 1d

225º 0x80483d4 - 1d

226º 0x80483d4 - 1d

227º 0x80483d4 - 1d

228º 0x80483d4 - 1d

229º 0x80483d4 - 1d

230º 0x80483d4 - 1d

231º 0x80483d4 - 1d

232º 0x80483d4 - 1d

233º 0x80483d4 - 1d

234º 0x80483d4 - 1d

235º 0x80483d4 - 1d

236º 0x80483d4 - 1d

237º 0x80483d4 - 1d

238º 0x80483d4 - 1d

239º 0x80483d4 - 1d

240º 0x80483d4 - 1d

241º 0x80483d4 - 1d

242º 0x80483d4 - 1d

243º 0x80483d4 - 1d

244º 0x80483d4 - 1d

245º 0x80483d4 - 1d

246º 0x80483d4 - 1d

247º 0x80483d4 - 1d

248º 0x80483d4 - 1d

249º 0x80483d4 - 1d

250º 0x80483d4 - 1d

251º 0x80483d4 - 1d

252º 0x80483d4 - 1d

253º 0x80483d4 - 1d

254º 0x80483d4 - 1d

255º 0x80483d4 - 1d

256º 0x80483d4 - 1d

257º 0x80483d4 - 1d

258º 0x80483d4 - 1d

259º 0x80483d4 - 1d

260º 0x80483d4 - 1d

261º 0x80483d4 - 1d

262º 0x80483d4 - 1d

263º 0x80483d4 - 1d

264º 0x80483d4 - 1d

265º 0x80483d4 - 1d

266º 0x80483d4 - 1d

267º 0x80483d4 - 1d

268º 0x80483d4 - 1d

269º 0x80483d4 - 1d

270º 0x80483d4 - 1d

271º 0x80483d4 - 1d

272º 0x80483d4 - 1d

273º 0x80483d4 - 1d

274º 0x80483d4 - 1d

275º 0x80483d4 - 1d

276º 0x80483d4 - 1d

277º 0x80483d4 - 1d

278º 0x80483d4 - 1d

279º 0x80483d4 - 1d

280º 0x80483d4 - 1d

281º 0x80483d4 - 1d

282º 0x80483d4 - 1d

283º 0x80483d4 - 1d

284º 0x80483d4 - 1d

285º 0x80483d4 - 1d

286º 0x80483d4 - 1d

287º 0x80483d4 - 1d

288º 0x80483d4 - 1d

289º 0x80483d4 - 1d

290º 0x80483d4 - 1d

291º 0x80483d4 - 1d

292º 0x80483d4 - 1d

293º 0x80483d4 - 1d

294º 0x80483d4 - 1d

295º 0x80483d4 - 1d

296º 0x80483d4 - 1d

297º 0x80483d4 - 1d

298º 0x80483d4 - 1d

299º 0x80483d4 - 1d

300º 0x80483d4 - 1d

301º 0x80483d4 - 1d

302º 0x80483d4 - 1d

303º 0x80483d4 - 1d

304º 0x80483d4 - 1d

305º 0x80483d4 - 1d

306º 0x80483d4 - 1d

307º 0x80483d4 - 1d

308º 0x80483d4 - 1d

309º 0x80483d4 - 1d

310º 0x80483d4 - 1d

311º 0x80483d4 - 1d

312º 0x80483d4 - 1d

313º 0x80483d4 - 1d

314º 0x80483d4 - 1d

315º 0x80483d4 - 1d

316º 0x80483d4 - 1d

317º 0x80483d4 - 1d

318º 0x80483d4 - 1d

319º 0x80483d4 - 1d

320º 0x80483d4 - 1d

321º 0x80483d4 - 1d

322º 0x80483d4 - 1d

323º 0x80483d4 - 1d

324º 0x80483d4 - 1d

325º 0x80483d4 - 1d

326º 0x80483d4 - 1d

327º 0x80483d4 - 1d

328º 0x80483d4 - 1d

329º 0x80483d4 - 1d

330º 0x80483d4 - 1d

331º 0x80483d4 - 1d

332º 0x80483d4 - 1d

333º 0x80483d4 - 1d

334º 0x80483d4 - 1d

335º 0x80483d4 - 1d

336º 0x80483d4 - 1d

337º 0x80483d4 - 1d

338º 0x80483d4 - 1d

339º 0x80483d4 - 1d

340º 0x80483d4 - 1d

341º 0x80483d4 - 1d

342º 0x80483d4 - 1d

343º 0x80483d4 - 1d

344º 0x80483d4 - 1d

345º 0x80483d4 - 1d

346º 0x80483d4 - 1d

347º 0x80483d4 - 1d

348º 0x80483d4 - 1d

349º 0x80483d4 - 1d

350º 0x80483d4 - 1d

351º 0x80483d4 - 1d

352º 0x80483d4 - 1d

353º 0x80483d4 - 1d

354º 0x80483d4 - 1d

355º 0x80483d4 - 1d

356º 0x80483d4 - 1d

357º 0x80483d4 - 1d

358º 0x80483d4 - 1d

359º 0x80483d4 - 1d

360º 0x80483d4 - 1d

361º 0x80483d4 - 1d

362º 0x80483d4 - 1d

363º 0x80483d4 - 1d

364º 0x80483d4 - 1d

365º 0x80483d4 - 1d

366º 0x80483d4 - 1d

367º 0x80483d4 - 1d

368º 0x80483d4 - 1d

369º 0x80483d4 - 1d

370º 0x80483d4 - 1d

371º 0x80483d4 - 1d

372º 0x80483d4 - 1d

373º 0x80483d4 - 1d

374º 0x80483d4 - 1d

375º 0x80483d4 - 1d

376º 0x80483d4 - 1d

377º 0x80483d4 - 1d

378º 0x80483d4 - 1d

379º 0x80483d4 - 1d

380º 0x80483d4 - 1d

381º 0x80483d4 - 1d

382º 0x80483d4 - 1d

383º 0x80483d4 - 1d

384º 0x80483d4 - 1d

385º 0x80483d4 - 1d

386º 0x80483d4 - 1d

387º 0x80483d4 - 1d

388º 0x80483d4 - 1d

389º 0x80483d4 - 1d

390º 0x80483d4 - 1d

391º 0x80483d4 - 1d

392º 0x80483d4 - 1d

393º 0x80483d4 - 1d

394º 0x80483d4 - 1d

395º 0x80483d4 - 1d

396º 0x80483d4 - 1d

397º 0x80483d4 - 1d

398º 0x80483d4 - 1d

399º 0x80483d4 - 1d

400º 0x80483d4 - 1d

401º 0x80483d4 - 1d

402º 0x80483d4 - 1d

403º 0x80483d4 - 1d

404º 0x80483d4 - 1d

405º 0x80483d4 - 1d

406º 0x80483d4 - 1d

407º 0x80483d4 - 1d

408º 0x80483d4 - 1d

409º 0x80483d4 - 1d

410º 0x80483d4 - 1d

411º 0x8

8º Nota: Ideal não vai à memória

iml (...) → não dáis acessos à mem.

add (...) → não dáis acessos à mem.

Número de acessos à mem. em -00:

- pushl %ebp;
- movl (%eax, %edx), %eax;
- pushl %eax;
- movl \$0, -8(%ebp);
- movl -8(%ebp), %eax;
- addl \$2(%ebp), %eax;
- movl \$2(%ebp), %eax;
- movl -8(%ebp), %eax;
- movl 8(%ebp), %eax;
- movl 8(%ebp), %eax;
- movl (%eax, %edx), %eax;
- movl 8(%ebp), %eax;
- movl %eax, (%eax, %edx);

Número de acessos à memória em -02:

- pushl %ebp;
- pushl %eax;
- movl 12(%ebp), %eax;
- movl 8(%ebp), %eax;
- movl (%eax, %edx), %eax;
- movl %eax, (%eax, %edx);
- hlt %eax;
- leave;
- ret

$$P_{0.8} 19 - q = 10$$

10. É viável garantir 8 algarismos significativos na representação de variáveis do tipo float? Para garantir que qualquer valor do tipo float em posição nómadas tem sempre pelo menos 8 algarismos significativos (mais 10), verifica-se necessário que a sua codificação em binário (em IEEE-754) permitisse representar pelo menos  $10^8$  valores diferentes. Sabendo que a codificação em ponto flutuante nómadas usa 32 bits, dos quais 23+1 são usados para a mantissa, então com esta notação apenas se podem representar  $2^{24}$  valores diferentes,  $\approx 16 \times 10^6$ , o que é claramente inferior a  $10^8$ . Temos de representar estes valores como double

19 acessos à memória

→ iml(%eax) x2

→ pushl %eax

→ leave

→ ret

\*

RISC → instruções nómadas e mais eficientes; operandas sempre em registo; formato nómadas de instruções, comprimento fixo e pausas variáveis.

q acessos

9º void para-hor (int a[], int m, int \*num){

int \*i;

int my-num = 0; → criar enda nova variável  
para cada entrar constantemente  
for (i = a, i < &a[m]; i++) → a ocdeia à memória

my-num += i; 3

(\*num) = my-num; 3

and O<sub>1</sub>, O<sub>2</sub> → novi-

fica no seu nú-

mero é par

(and \$1, %eax)

para-hor(a, 5, &b);

int O<sub>1</sub>, O<sub>2</sub> → armazenar

o os "and" mas

mete guarda o valor em

O<sub>2</sub>.

float (%eax, %edx, 2) → 8array[i];

movl (%eax, %edx, 2) → array[i];

and O<sub>1</sub>, O<sub>2</sub> → ou exclusivo (igual a

zero se O<sub>1</sub> = O<sub>2</sub>)

(algumas instruções importantes)