

初赛阶段性总结_lyz

一、前言

PA (Programming Assignment) 是南京大学计算机系统基础课程的小型项目。旨在让学生从“零”开始实现一个完整的计算机系统。而本项目旨在实现南大PA实验的 LoongArch 支持，团队初级阶段所完成的任务如下：基础设施实现与测试、nemu 的LA实现以及 Loongarch32r 适配等。

二、实现部分

(1) 个人工作简述

1.1 表达式求值的实现与测试

表达式求值的目标是创建一个能够解析和求值复杂数学表达式的工具，能够处理包含寄存器名和十六进制数的表达式，并将其求值结果根据情况以十进制或十六进制的格式进行输出。简单实现效果如下：

```
1 (nemu) p 125*8 + 3
2 125*8 + 3 = 1003
3
4 (nemu) p $t0 + $pc
5 $t0 + $pc = 0x80000000
6
7 (nemu) p 0x1000000 + $pc
8 0x1000000 + $pc = 0x81000000
```

为了确保求值工具能够正确处理各种情况，因此采用随机生成表达式函数来生成一系列测试表达式，并将这些表达式输入到我们的表达式求值工具中进行求值。通过比较表达式求值工具输出的结果与预期结果，便能够判断这个工具的准确性和稳定性。

生成的随机表达式

测试结果

```
nemu > tools > gen-expr > input
63 189 97+(92)
64 2237 (((66*34)+74)-(81))
65 -30 ((55/39+(77)-21-(((87))))))
66 82 82
67 45 45
68 60 76*(57)/72
69 69846 (((73/(1))*((76))-(46)+((24))+((60))*16*(67)))
70 16 (81/57/44/24)+16
71 -37 93-75-55+(17/44/70)*30
72 -2288 68+3-(2336)-(34/90)*(50-965838-38)-(51/19/7570+23)
73 -4906 ((49*24/48+(6-(91))*((17+(28)+83/6)))
74 135663 135663
75 -68 6438/71+64-(28)*(((15))))+3*66
76 19684 6438/71+64-(28)*(((15))))+3*6650
77 92 (((58))+34)
78 189 97+(92)
79 69846 (((73/(1))*((76))-(46)+((24))+((60))*16*(67)))
80 -121 (25)-(90)-(((56))-(((29)/67/((5))/(((47)-(33))))))
```

```
nemu > tools > gen-expr > test.c
268 while (fgets(line, sizeof(line), file) != NULL) {
269     long result;
270     char expression[256];
271     if (sscanf(line, "%ld %255s", &result, expression) == 2) {
272         f_len++;
273         bool success = true;
274         init_regex();
275         long res = expr(expression, &success);
276         if (!success) {
277             puts("invalid expression!");
278         } else {
279             assert(result == res);
280         }
281     } else {
282         printf("Invalid line: %s", line);
283     }
284 }
285 fclose(file);
```

1.2 Klib实现难点与测试

实现目标

- 内存和字符串的写入函数, 例如 `memset()`, `strcpy()` 等。
- 内存和字符串的只读函数, 例如 `memcmp()`, `strlen()` 等。
- 格式化输出函数, 例如 `sprintf()` 等。

遇到难点

- 对于c程序的部分头文件, 如 `<stdarg.h>` 等不太熟悉, 因此一开始实现的时候碰壁加钻牛角尖了, 后续了解到有此头文件后, 通过利用其包含的宏即可更好地接收与处理可变参数。

测试思路

1. **随机生成字符串**: 利用随机测试法, 通过代码生产随机长度及内容的字符串用于测试。

```
1 // 生成随机字符串的函数
2 void generate_random_string(char *str, size_t min_length, size_t max_length) {
3     const char charset[] =
4         "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 ";
5     size_t charset_size = sizeof(charset) - 1;
6     size_t length = min_length + rand() % (max_length - min_length + 1);
7     for (size_t i = 0; i < length; ++i) {
8         str[i] = charset[rand() % charset_size];
9     }
10
11     str[length] = '\0'; // 添加字符串结束符
12 }
```

2. **内存和字符串的写入函数测试**: 对于内存和字符串的写入函数, 如 `memset()`、`strcpy()` 等, 我们可以通过构建特定的测试场景来验证函数的行为。例如, 对于 `memset()`, 我们可以使用辅助函数来检查函数对指定内存区域的写入效果。对于 `strcpy()`, 我们可以检查源字符串和目标字符串之间的字符复制情况。

3. **内存和字符串的只读函数测试**：对于内存和字符串的只读函数，如 `memcmp()`、`strlen()` 等，我们需要构造能够产生预期输出的测试场景。对于 `memcmp()`，我们可以通过比较两个内存块的前n个字节来测试函数的行为。对于 `strlen()`，我们可以通过测量字符串的长度来验证函数的准确性。
4. **格式化输出函数测试**：对于格式化输出函数，如 `sprintf()` 等，我们需要构造具有代表性的输入数据，并通过比较实际输出与预期输出来验证函数的行为。这里可以考虑使用 `printf()` 函数来生成预期的输出字符串，然后与实际输出进行比较。

测试效果

stdio-test测试效果

```
proj241-PA_for_LA > am-kernels > tests > klib-test > C klib-stdio-test.c > main()
100 const size_t max_string_length = 50; // 设置最大字符串长度
101 char random_string[max_string_length + 1];
102 char test[max_string_length + 1];
103 int test_count = 0;
104
105 for (int i = 0; i < 10000; i++) {
106     generate_random_string(random_string, min_string_length, max_string_length);
107
108     // 使用sprintf函数将随机字符串格式化输出到test字符串中
109     sprintf(test, "%s", random_string);
110
111     // 检查输出是否与原始字符串匹配
112     if (strcmp(test, random_string)) {
113         printf("Test failed at iteration %d\n", test_count);
114         return -1;
115     }
116
117     test_count++;
118 }
119
120 printf("All tests passed successfully.\n");
121 return 0;
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

string-test测试效果（并随机输出一部分字符串）

```
proj241-PA_for_LA > am-kernels > tests > klib-test > C klib-string-test.c > main()
180 // == 测试 memcpy 和 memmove == //
181 memcpy1(memcpy_test, random_string, sizeof(random_string));
182 if (memcmp1(memcpy_test, random_string, sizeof(random_string)) != 0)
183     return -1;
184
185 // == 测试 memmove == //
186 strcpyl(memmove_test, random_string);
187 strcpyl(memmove_test1, random_string);
188 memmove1(memmove_test+1, memmove_test, str_len/2);
189 memmove(memmove_test1+1, memmove_test1, str_len/2);
190 if (strcmp(memmove_test, memmove_test1))
191     return -1;
192
193 // == 测试 memset == //
194 // 清空数组
195 memset1(random_string, 0, sizeof(random_string));
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

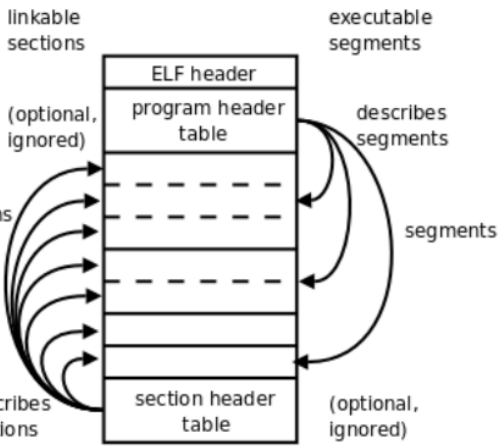
1.3 ELF学习文档及ftrace工具的实现

①ELF学习

对ELF文件格式进行学习，以便于ftrace工具的实现。了解到是一种通用的二进制文件格式，用于在Unix类操作系统中存储可执行程序、共享库和目标文件，是编译器和链接器生成的，其中包含了源代码编译后的机器代码、全局变量、符号表、重定位表等信息。

从ELF文件的格式图（右图）可知ELF文件由以下五大部分构成：ELF Header（ELF文件头）、程序头表（Program Header Table）、节头表（Section Header Table）、节（Sections）、符号表（Symbol Table）

关于 ELF 的详细资料请看[ELF学习文档](#)。



②ftrace实现

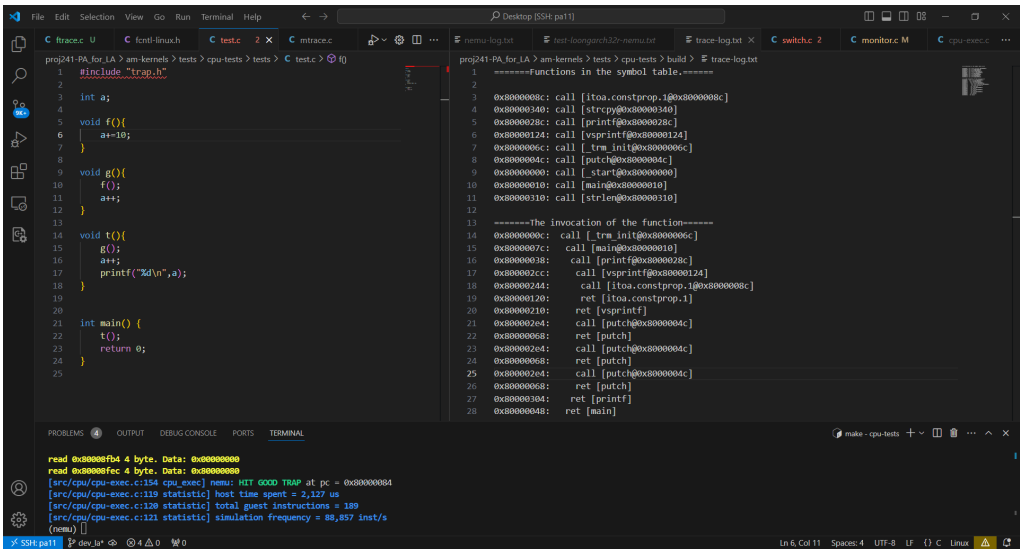
实现目标

- 从符号表中解析出函数的名称（Name）以及地址（Value）
- 解析出 `call` 指令和 `ret` 指令的执行地址
- 解析调用关系并记录

实现思路

- 1、从符号表中解析出函数的名称（Name）以及地址（Value）
- 2、在程序执行过程中将call指令与ret指令地址存储
- 3、处理函数调用关系及返回关系

实现效果



(2) 团队协作

1.1 日常讨论

在初赛阶段的团队协作中，队员之间频繁的进行讨论，既是互相学习，也是推动项目发展的重要方式。通过交流学习，面对同一个问题各自不同的看法均可互相借鉴，以寻求最有效的解决方式。

1.2 定期会议与更新

团队定期举行会议，平均每两周一次，确保每个团队成员都能了解项目的最新进展、遇到的问题以及解决方案。这些会议不仅是分享工作成果的机会，也是解决问题和分工协调的关键环节。通过会议，我们能够及时调整计划，确保项目按时完成。

在分享会议中，既可以同步所有成员的项目进度，也可以互相发现问题并将其解决。在互相学习、交流帮助的环境下，更加有助于每个人的成长。

1.3 分工明确与责任分配

为了提高效率，队长为我们每个队员的职责进行了分工。每个人都负责一个或几个特定的模块或功能，这样可以减少冲突，避免重复工作，提高工作效率。同时建立了任务甘特图，让每个人能够看到团队项目的任务状态，以及团队整体的进度。

三、阶段感想与总结

本人在初赛阶段所做的主要任务为基础设施的实现与测试、文档编写，与团队成员协作完成 `nemu` 实现等工作。通过次初级阶段，也让我收获颇多。

- **技术能力的提升：**通过实现基础设施和测试的过程中，我对项目相关的技术与原理有了更多的认识，通过nemu实现的过程，也在一步步提高我对整个计算机系统的理解。同时，也掌握了多种代码及工具的测试方式，始终牢记“机器永远是对的”、“未测试的代码永远是错的”两大原则，进一步提升代码编写能力与工程开发思维。
- **团队协作能力的增强：**与团队成员一起完成项目，我学会了如何更有效地与他人合作，如何协调不同的意见和想法，以及如何共同解决问题。同时也是真正意义上的感觉到团队协作的重要性，与定期与团队成员进行小阶段成果分享与讨论，将每个人所做的部分联系起来，既有助于个人对整个项目的理解，又能拾取到个人没注意到的小细节问题，更有利于团队开发进度的推进。
- **问题解决能力的提高：**在遇到问题时不应该是第一时间求助，而是必须先经过自己的思考与调试。最大的感想就是一句话：“有报错信息是好事”，至少我们可以知道是哪方面的报错信息，便可以针对性的处理问题，善于STFW（Search The Friendly Web，在友好的网络上搜索）、RTFM（Read The Friendly Manual，阅读友好的手册）、RTFSC（Read The Friendly Source Code，阅读友好的源代码）来帮助我们解决问题，发现问题后实在解决不了的，可以向他人求助，但一定要表述清楚问题以及自己的分析过程，以便对方更好地帮助我们。

总的来说，通过初赛阶段的参与，我不仅丰富了自己的技术储备和团队协作经验，也提升了自己的问题解决能力和工程开发思维。从最初的基础知识薄弱，到现在具有一定的知识面，这样的经历将对我未来的学习和职业生涯产生积极影响。

四、深入思考与自我反省

1. 反思

个人在项目开发过程中所承担的工作量其实并不多，更多的是进行项目分析与实现的工具测试。一部分原因是由于本身琐事太多，难以有效的利用时间来投入到项目中，因此为了不影响整个项目的进度，个人便减少了关于主线开发的任务认领。其实这样对于我而言所能学到的东西就少了很多，因此我也有很强的二周目意愿，一定要单独做完整个项目，去面对更多的问题，将其吸收解决，既是锻炼心态的过程，也是提高能力的最好途径。

2. 思考

个人觉得还有一部分原因是由于整个项目主线所占的开发比重较大，工具开发占比较小，因此一旦一名组员承担了主线的推进，另一名组员就必然难以跟上主线一起帮忙推进，为了避免重复工作，两名组员之间只能通过交流的方式共享进度，这样会减少相当一部分问题的发现与解决，并且实现细节部分也难以都共享清楚，因此，若要交接主线的推进，将会遇到很多上一位队员遇到的问题，导致工作重复并影响项目推进的效率。

五、未来规划

在后续，我将继续参与到项目的完善中，与队员一起共同完成整个项目，并进行项目代码的优化。

完成整个项目后，我也将探索并尝试完成那些在项目过程中未被分配到的部分。这不仅能帮助我进一步加深对整个项目的理解，也能促进我对计算机系统的深入学习。通过接触和解决这些新问题，我期待能够学习到更多的知识和技能，同时也能丰富我的项目经历。