

# PA1 问题记录

为了提高调试的效率,同时也作为熟悉框架代码的练习,我们需要在monitor中实现一个具有如下功能的简易调试器(相关部分的代码在 `nemu/src/monitor/debug/` 目录下),如果你不清楚命令的格式和功能,请参考如下表格:

命令	格式	使用举例	说明
帮助(1)	<code>help</code>	<code>help</code>	打印命令的帮助信息
继续运行(1)	<code>c</code>	<code>c</code>	继续运行被暂停的程序
退出(1)	<code>q</code>	<code>q</code>	退出NEMU
单步执行	<code>si [N]</code>	<code>si 10</code>	让程序单步执行 <code>N</code> 条指令后暂停执行,当 <code>N</code> 没有给出时,缺省为 <code>1</code>
打印程序状态	<code>info SUBCMD</code>	<code>info r</code> <code>info w</code>	打印寄存器状态 打印监视点信息
表达式求值	<code>p EXPR</code>	<code>p \$eax + 1</code>	求出表达式 <code>EXPR</code> 的值, <code>EXPR</code> 支持的运算请见 <a href="#">调试中的表达式求值</a> 小节
扫描内存(2)	<code>x N EXPR</code>	<code>x 10 \$esp</code>	求出表达式 <code>EXPR</code> 的值,将结果作为起始内存地址,以十六进制形式输出连续的 <code>N</code> 个4字节
设置监视点	<code>w EXPR</code>	<code>w *0x2000</code>	当表达式 <code>EXPR</code> 的值发生变化时,暂停程序执行
删除监视点	<code>d N</code>	<code>d 2</code>	删除序号为 <code>N</code> 的监视点

备注:

- (1) 命令已实现
- (2) 与GDB相比,我们在这里做了简化,更改了命令的格式

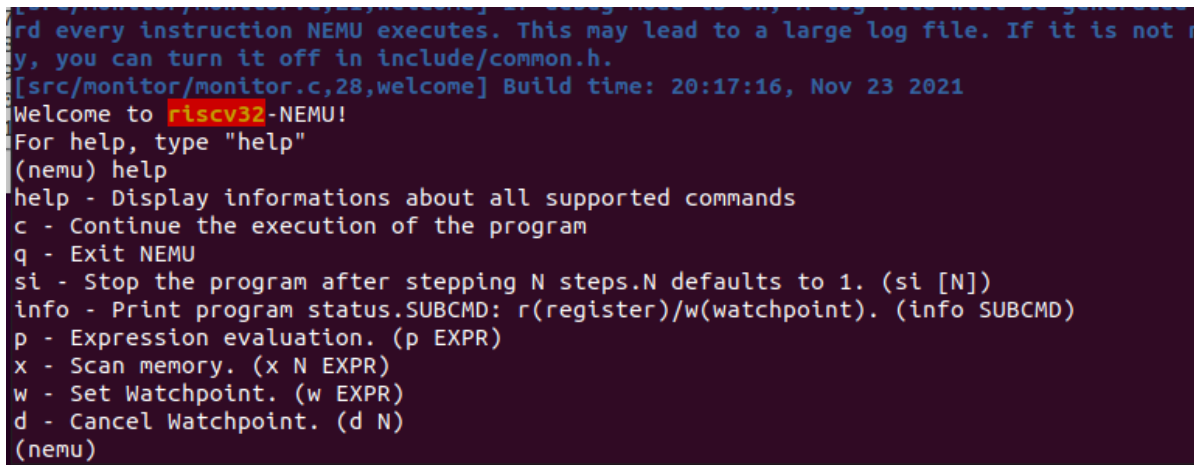
## 1. 帮助

每完成一条指令,都应在`cmd_table`中加入新指令的描述及对应操作符:



```
47 static int cmd_d(char *args);
48
49 static struct {
50     char *name;
51     char *description;
52     int (*handler) (char *);
53 } cmd_table [] = {
54     { "help", "Display informations about all supported commands", cmd_help },
55     { "c", "Continue the execution of the program", cmd_c },
56     { "q", "Exit NEMU", cmd_q },
57     { "si", "Stop the program after stepping N steps.N defaults to 1. (si [N])", cmd_si},
58     { "info", "Print program status.SUBCMD: r(register)/w(watchpoint). (info SUBCMD)", cmd_info},
59     { "p", "Expression evaluation. (p EXPR)", cmd_p},
60     { "x", "Scan memory. (x N EXPR)", cmd_x},
61     { "w", "Set Watchpoint. (w EXPR)", cmd_w},
62     { "d", "Cancel Watchpoint. (d N)", cmd_d},
63     /* TODO: Add more commands */
64
65 };
66
```

包含所有指令的截图如下：



```
rd every instruction NEMU executes. This may lead to a large log file. If it is not r
y, you can turn it off in include/common.h.
[src/monitor/monitor.c,28,welcome] Build time: 20:17:16, Nov 23 2021
Welcome to riscv32-NEMU!
For help, type "help"
(nemu) help
help - Display informations about all supported commands
c - Continue the execution of the program
q - Exit NEMU
si - Stop the program after stepping N steps.N defaults to 1. (si [N])
info - Print program status.SUBCMD: r(register)/w(watchpoint). (info SUBCMD)
p - Expression evaluation. (p EXPR)
x - Scan memory. (x N EXPR)
w - Set Watchpoint. (w EXPR)
d - Cancel Watchpoint. (d N)
(nemu)
```

## 2. 单步执行

指令函数位置：

```
nemu/src/monitor/debug/ui.c
```

代码如下：

```

80
81 static int cmd_si(char *args) {
82     /* extract the first argument */
83     char *arg = strtok(NULL, " ");
84     char *endptr = "";
85     long N;
86
87     if(arg == NULL) {
88         /* N is not given */
89         N = 1;
90     } else {
91         /* get the Num */
92         N = strtol(arg, &endptr, 10);
93     }
94
95     if(strlen(endptr)>0){
96         /* if enter error */
97         printf("Incorrect format, please re-enter the instructions like \"si [N]\"\n");
98     } else {
99         cpu_exec(N);
100     }
101
102     return 0;
103 }
104

```

在只输入si时默认执行1条指令；si后面如果输入数字的格式不对则会触发提示。执行结果如下：

```

Welcome to riscv32-NEMU!
For help, type "help"
(nemu) si
80100000: b7 02 00 80          lui 0x80000,t0
(nemu) si asd
Incorrect format, please re-enter the instructions like "si [N]"
(nemu) si 8id1
Incorrect format, please re-enter the instructions like "si [N]"
(nemu) si 10
80100008: 03 a5 02 00          lw 0(t0),a0
8010000c: 6b 00 00 00          nemu trap
nemu: HIT GOOD TRAP at pc = 0x8010000c

[src/monitor/cpu-exec.c,29,monitor_statistic] total guest instructions = 4
(nemu)

```

### 3. 打印程序状态

指令函数位置：

```
nemu/src/monitor/debug/ui.c
```

函数代码：

```
16 static int cmd_info(char *args) {
17     /* extract the first argument */
18     char *arg = strtok(NULL, " ");
19
20     if(strcmp(arg, "r") == 0){
21         /*print register status*/
22         isa_reg_display();
23     }else if(strcmp(arg, "w") == 0){
24         /*print watchpoint info*/
25         wp_display();
26     }else{
27         /*if enter error*/
28         printf("Incorrect format, please re-enter the instructions like \"info SUBCMD\"\n");
29     }
30
31     return 0;
32 }
```

下面是isa\_reg\_display()和wp\_display()两个函数的过程：

```
info r //isa_reg_display()
```

通过nemi/src/isa/riscv32/include/isa/reg.h头文件可以找到reg\_name函数，这个函数可以按规定宽度打印出对应寄存器名。

对应寄存器状态的函数则在reg\_l中被定义。如下图：



```
1 打开(O)  x  reg.h  保存(S)  -  □
2  ~./hust/ics2019/nemu/src/isa/riscv32/include/isa
3
4  *.ui.c  x  reg.c  reg.h
5
6  4 #include "common.h"
7
8  5 #define PC_START (0x80000000u + IMAGE_START)
9
10 6
11 7 typedef struct {
12 8     struct {
13 9         rtlreg_t _32;
14 10     } gpr[32];
15 11
16 12     vaddr_t pc;
17 13
18 14 } CPU_state;
19
20 15
21 16 static inline int check_reg_index(int index) {
22 17     assert(index >= 0 && index < 32);
23 18     return index;
24 19 }
25
26 1
27 2 #define reg_l(index) (cpu.gpr[check_reg_index(index)]._32)
28 3
29 4 static inline const char* reg_name(int index, int width) {
30 5     extern const char* regsl[];
31 6     assert(index >= 0 && index < 32);
32 7     return regsl[index];
33 8 }
34
35 9
36 10
37 11
38 12
39 13
40 14
41 15
42 16
43 17
44 18
45 19
46 20
47 21
48 22
49 23
50 24
51 25
52 26
53 27
54 28
55 29
56 30
57 31
58 32
59 33
60 34
61 35
62 36
63 37
64 38
65 39
66 40
67 41
68 42
69 43
70 44
71 45
72 46
73 47
74 48
75 49
76 50
77 51
78 52
79 53
80 54
81 55
82 56
83 57
84 58
85 59
86 60
87 61
88 62
89 63
90 64
91 65
92 66
93 67
94 68
95 69
96 70
97 71
98 72
99 73
100 74
101 75
102 76
103 77
104 78
105 79
106 80
107 81
108 82
109 83
110 84
111 85
112 86
113 87
114 88
115 89
116 90
117 91
118 92
119 93
120 94
121 95
122 96
123 97
124 98
125 99
126 100
127 101
128 102
129 103
130 104
131 105
132 106
133 107
134 108
135 109
136 110
137 111
138 112
139 113
140 114
141 115
142 116
143 117
144 118
145 119
146 120
147 121
148 122
149 123
150 124
151 125
152 126
153 127
154 128
155 129
156 130
157 131
158 132
159 133
160 134
161 135
162 136
163 137
164 138
165 139
166 140
167 141
168 142
169 143
170 144
171 145
172 146
173 147
174 148
175 149
176 150
177 151
178 152
179 153
180 154
181 155
182 156
183 157
184 158
185 159
186 160
187 161
188 162
189 163
190 164
191 165
192 166
193 167
194 168
195 169
196 170
197 171
198 172
199 173
200 174
201 175
202 176
203 177
204 178
205 179
206 180
207 181
208 182
209 183
210 184
211 185
212 186
213 187
214 188
215 189
216 190
217 191
218 192
219 193
220 194
221 195
222 196
223 197
224 198
225 199
226 200
227 201
228 202
229 203
230 204
231 205
232 206
233 207
234 208
235 209
236 210
237 211
238 212
239 213
240 214
241 215
242 216
243 217
244 218
245 219
246 220
247 221
248 222
249 223
250 224
251 225
252 226
253 227
254 228
255 229
256 230
257 231
258 232
259 233
260 234
261 235
262 236
263 237
264 238
265 239
266 240
267 241
268 242
269 243
270 244
271 245
272 246
273 247
274 248
275 249
276 250
277 251
278 252
279 253
280 254
281 255
282 256
283 257
284 258
285 259
286 260
287 261
288 262
289 263
290 264
291 265
292 266
293 267
294 268
295 269
296 270
297 271
298 272
299 273
300 274
301 275
302 276
303 277
304 278
305 279
306 280
307 281
308 282
309 283
310 284
311 285
312 286
313 287
314 288
315 289
316 290
317 291
318 292
319 293
320 294
321 295
322 296
323 297
324 298
325 299
326 300
327 301
328 302
329 303
330 304
331 305
332 306
333 307
334 308
335 309
336 310
337 311
338 312
339 313
340 314
341 315
342 316
343 317
344 318
345 319
346 320
347 321
348 322
349 323
350 324
351 325
352 326
353 327
354 328
355 329
356 330
357 331
358 332
359 333
360 334
361 335
362 336
363 337
364 338
365 339
366 340
367 341
368 342
369 343
370 344
371 345
372 346
373 347
374 348
375 349
376 350
377 351
378 352
379 353
380 354
381 355
382 356
383 357
384 358
385 359
386 360
387 361
388 362
389 363
390 364
391 365
392 366
393 367
394 368
395 369
396 370
397 371
398 372
399 373
400 374
401 375
402 376
403 377
404 378
405 379
406 380
407 381
408 382
409 383
410 384
411 385
412 386
413 387
414 388
415 389
416 390
417 391
418 392
419 393
420 394
421 395
422 396
423 397
424 398
425 399
426 400
427 401
428 402
429 403
430 404
431 405
432 406
433 407
434 408
435 409
436 410
437 411
438 412
439 413
440 414
441 415
442 416
443 417
444 418
445 419
446 420
447 421
448 422
449 423
450 424
451 425
452 426
453 427
454 428
455 429
456 430
457 431
458 432
459 433
460 434
461 435
462 436
463 437
464 438
465 439
466 440
467 441
468 442
469 443
470 444
471 445
472 446
473 447
474 448
475 449
476 450
477 451
478 452
479 453
480 454
481 455
482 456
483 457
484 458
485 459
486 460
487 461
488 462
489 463
490 464
491 465
492 466
493 467
494 468
495 469
496 470
497 471
498 472
499 473
500 474
501 475
502 476
503 477
504 478
505 479
506 480
507 481
508 482
509 483
510 484
511 485
512 486
513 487
514 488
515 489
516 490
517 491
518 492
519 493
520 494
521 495
522 496
523 497
524 498
525 499
526 500
527 501
528 502
529 503
530 504
531 505
532 506
533 507
534 508
535 509
536 510
537 511
538 512
539 513
540 514
541 515
542 516
543 517
544 518
545 519
546 520
547 521
548 522
549 523
550 524
551 525
552 526
553 527
554 528
555 529
556 530
557 531
558 532
559 533
560 534
561 535
562 536
563 537
564 538
565 539
566 540
567 541
568 542
569 543
570 544
571 545
572 546
573 547
574 548
575 549
576 550
577 551
578 552
579 553
580 554
581 555
582 556
583 557
584 558
585 559
586 560
587 561
588 562
589 563
590 564
591 565
592 566
593 567
594 568
595 569
596 570
597 571
598 572
599 573
600 574
601 575
602 576
603 577
604 578
605 579
606 580
607 581
608 582
609 583
610 584
611 585
612 586
613 587
614 588
615 589
616 590
617 591
618 592
619 593
620 594
621 595
622 596
623 597
624 598
625 599
626 600
627 601
628 602
629 603
630 604
631 605
632 606
633 607
634 608
635 609
636 610
637 611
638 612
639 613
640 614
641 615
642 616
643 617
644 618
645 619
646 620
647 621
648 622
649 623
650 624
651 625
652 626
653 627
654 628
655 629
656 630
657 631
658 632
659 633
660 634
661 635
662 636
663 637
664 638
665 639
666 640
667 641
668 642
669 643
670 644
671 645
672 646
673 647
674 648
675 649
676 650
677 651
678 652
679 653
680 654
681 655
682 656
683 657
684 658
685 659
686 660
687 661
688 662
689 663
690 664
691 665
692 666
693 667
694 668
695 669
696 670
697 671
698 672
699 673
700 674
701 675
702 676
703 677
704 678
705 679
706 680
707 681
708 682
709 683
710 684
711 685
712 686
713 687
714 688
715 689
716 690
717 691
718 692
719 693
720 694
721 695
722 696
723 697
724 698
725 699
726 700
727 701
728 702
729 703
730 704
731 705
732 706
733 707
734 708
735 709
736 710
737 711
738 712
739 713
740 714
741 715
742 716
743 717
744 718
745 719
746 720
747 721
748 722
749 723
750 724
751 725
752 726
753 727
754 728
755 729
756 730
757 731
758 732
759 733
760 734
761 735
762 736
763 737
764 738
765 739
766 740
767 741
768 742
769 743
770 744
771 745
772 746
773 747
774 748
775 749
776 750
777 751
778 752
779 753
780 754
781 755
782 756
783 757
784 758
785 759
786 760
787 761
788 762
789 763
790 764
791 765
792 766
793 767
794 768
795 769
796 770
797 771
798 772
799 773
800 774
801 775
802 776
803 777
804 778
805 779
806 780
807 781
808 782
809 783
810 784
811 785
812 786
813 787
814 788
815 789
816 790
817 791
818 792
819 793
820 794
821 795
822 796
823 797
824 798
825 799
826 800
827 801
828 802
829 803
830 804
831 805
832 806
833 807
834 808
835 809
836 810
837 811
838 812
839 813
840 814
841 815
842 816
843 817
844 818
845 819
846 820
847 821
848 822
849 823
850 824
851 825
852 826
853 827
854 828
855 829
856 830
857 831
858 832
859 833
860 834
861 835
862 836
863 837
864 838
865 839
866 840
867 841
868 842
869 843
870 844
871 845
872 846
873 847
874 848
875 849
876 850
877 851
878 852
879 853
880 854
881 855
882 856
883 857
884 858
885 859
886 860
887 861
888 862
889 863
890 864
891 865
892 866
893 867
894 868
895 869
896 870
897 871
898 872
899 873
900 874
901 875
902 876
903 877
904 878
905 879
906 880
907 881
908 882
909 883
910 884
911 885
912 886
913 887
914 888
915 889
916 890
917 891
918 892
919 893
920 894
921 895
922 896
923 897
924 898
925 899
926 900
927 901
928 902
929 903
930 904
931 905
932 906
933 907
934 908
935 909
936 910
937 911
938 912
939 913
940 914
941 915
942 916
943 917
944 918
945 919
946 920
947 921
948 922
949 923
950 924
951 925
952 926
953 927
954 928
955 929
956 930
957 931
958 932
959 933
960 934
961 935
962 936
963 937
964 938
965 939
966 940
967 941
968 942
969 943
970 944
971 945
972 946
973 947
974 948
975 949
976 950
977 951
978 952
979 953
980 954
981 955
982 956
983 957
984 958
985 959
986 960
987 961
988 962
989 963
990 964
991 965
992 966
993 967
994 968
995 969
996 970
997 971
998 972
999 973
1000 974
1001 975
1002 976
1003 977
1004 978
1005 979
1006 980
1007 981
1008 982
1009 983
1010 984
1011 985
1012 986
1013 987
1014 988
1015 989
1016 990
1017 991
1018 992
1019 993
1020 994
1021 995
1022 996
1023 997
1024 998
1025 999
1026 1000
1027 1001
1028 1002
1029 1003
1030 1004
1031 1005
1032 1006
1033 1007
1034 1008
1035 1009
1036 1010
1037 1011
1038 1012
1039 1013
1040 1014
1041 1015
1042 1016
1043 1017
1044 1018
1045 1019
1046 1020
1047 1021
1048 1022
1049 1023
1050 1024
1051 1025
1052 1026
1053 1027
1054 1028
1055 1029
1056 1030
1057 1031
1058 1032
1059 1033
1060 1034
1061 1035
1062 1036
1063 1037
1064 1038
1065 1039
1066 1040
1067 1041
1068 1042
1069 1043
1070 1044
1071 1045
1072 1046
1073 1047
1074 1048
1075 1049
1076 1050
1077 1051
1078 1052
1079 1053
1080 1054
1081 1055
1082 1056
1083 1057
1084 1058
1085 1059
1086 1060
1087 1061
1088 1062
1089 1063
1090 1064
1091 1065
1092 1066
1093 1067
1094 1068
1095 1069
1096 1070
1097 1071
1098 1072
1099 1073
1100 1074
1101 1075
1102 1076
1103 1077
1104 1078
1105 1079
1106 1080
1107 1081
1108 1082
1109 1083
1110 1084
1111 1085
1112 1086
1113 1087
1114 1088
1115 1089
1116 1090
1117 1091
1118 1092
1119 1093
1120 1094
1121 1095
1122 1096
1123 1097
1124 1098
1125 1099
1126 1100
1127 1101
1128 1102
1129 1103
1130 1104
1131 1105
1132 1106
1133 1107
1134 1108
1135 1109
1136 1110
1137 1111
1138 1112
1139 1113
1140 1114
1141 1115
1142 1116
1143 1117
1144 1118
1145 1119
1146 1120
1147 1121
1148 1122
1149 1123
1150 1124
1151 1125
1152 1126
1153 1127
1154 1128
1155 1129
1156 1130
1157 1131
1158 1132
1159 1133
1160 1134
1161 1135
1162 1136
1163 1137
1164 1138
1165 1139
1166 1140
1167 1141
1168 1142
1169 1143
1170 1144
1171 1145
1172 1146
1173 1147
1174 1148
1175 1149
1176 1150
1177 1151
1178 1152
1179 1153
1180 1154
1181 1155
1182 1156
1183 1157
1184 1158
1185 1159
1186 1160
1187 1161
1188 1162
1189 1163
1190 1164
1191 1165
1192 1166
1193 1167
1194 1168
1195 1169
1196 1170
1197 1171
1198 1172
1199 1173
1200 1174
1201 1175
1202 1176
1203 1177
1204 1178
1205 1179
1206 1180
1207 1181
1208 1182
1209 1183
1210 1184
1211 1185
1212 1186
1213 1187
1214 1188
1215 1189
1216 1190
1217 1191
1218 1192
1219 1193
1220 1194
1221 1195
1222 1196
1223 1197
1224 1198
1225 1199
1226 1200
1227 1201
1228 1202
1229 1203
1230 1204
1231 1205
1232 1206
1233 1207
1234 1208
1235 1209
1236 1210
1237 1211
1238 1212
1239 1213
1240 1214
1241 1215
1242 1216
1243 1217
1244 1218
1245 1219
1246 1220
1247 1221
1248 1222
1249 1223
1250 1224
1251 1225
1252 1226
1253 1227
1254 1228
1255 1229
1256 1230
1257 1231
1258 1232
1259 1233
1260 1234
1261 1235
1262 1236
1263 1237
1264 1238
1265 1239
1266 1240
1267 1241
1268 1242
1269 1243
1270 1244
1271 1245
1272 1246
1273 1247
1274 1248
1275 1249
1276 1250
1277 1251
1278 1252
1279 1253
1280 1254
1281 1255
1282 1256
1283 1257
1284 1258
1285 1259
1286 1260
1287 1261
1288 1262
1289 1263
1290 1264
1291 1265
1292 1266
1293 1267
1294 1268
1295 1269
1296 1270
1297 1271
1298 1272
1299 1273
1300 1274
1301 1275
1302 1276
1303 1277
1304 1278
1305 1279
1306 1280
1307 1281
1308 1282
1309 1283
1310 1284
1311 1285
1312 1286
1313 1287
1314 1288
1315 1289
1316 1290
1317 1291
1318 1292
1319 1293
1320 1294
1321 1295
1322 1296
1323 1297
1324 1298
1325 1299
1326 1300
1327 1301
1328 1302
1329 1303
1330 1304
1331 1305
1332 1306
1333 1307
1334 1308
1335 1309
1336 1310
1337 1311
1338 1312
1339 1313
1340 1314
1341 1315
1342 1316
1343 1317
1344 1318
1345 1319
1346 1320
1347 1321
1348 1322
1349 1323
1350 1324
1351 1325
1352 1326
1353 1327
1354 1328
1355 1329
1356 1330
1357 1331
1358 1332
1359 1333
1360 1334
1361 1335
1362 1336
1363 1337
1364 1338
1365 1339
1366 1340
1367 1341
1368 1342
1369 1343
1370 1344
1371 1345
1372 1346
1373 1347
1374 1348
1375 1349
1376 1350
1377 1351
1378 1352
1379 1353
1380 1354
1381 1355
1382 1356
1383 1357
1384 1358
1385 1359
1386 1360
1387 1361
1388 1362
1389 1363
1390 1364
1391 1365
1392 1366
1393 1367
1394 1368
1395 1369
1396 1370
1397 1371
1398 1372
1399 1373
1400 1374
1401 1375
1402 1376
1403 1377
1404 1378
1405 1379
1406 1380
1407 1381
1408 1382
1409 1383
1410 1384
1411 1385
1412 1386
1413 1387
1414 1388
1415 1389
1416 1390
1417 1391
1418 1392
1419 1393
1420 1394
1421 1395
1422 1396
1423 1397
1424 1398
1425 1399
1426 1400
1427 1401
1428 1402
1429 1403
1430 1404
1431 1405
1432 1406
143
```

## 4. 表达式求值

按照给的框架写eval函数

### PA1 - 开天辟地的篇章: 最简单的计算机

在开始愉快的PA之旅之前

开天辟地的篇章

RTFSC

基础设施

表达式求值

监视点

如何阅读手册

### PA2 - 简单复杂的机器: 冯诺依曼计算...

不停计算的机器

RTFSC(2)

程序, 运行时环境与AM

基础设施(2)

输入输出

### PA3 - 穿越时空的旅程: 批处理系统

最简单的操作系统

穿越时空的旅程

用户程序和系统调用

文件系统

批处理系统

### PA4 - 虚实交错的魔法: 分时多任务

多道程序

虚实交错的魔法

超越容量的界限

找到了正确的主运算符之后, 事情就变得很简单了: 先对分裂出来的两个子表达式进行递归求值, 然后再根据主运算符的类型对两个子表达式的值进行运算即可。完整的求值函数如下:

```
eval(p, q) {
  if (p > q) {
    /* Bad expression */
  }
  else if (p == q) {
    /* Single token.
     * For now this token should be a number.
     * Return the value of the number.
     */
  }
  else if (check_parentheses(p, q) == true) {
    /* The expression is surrounded by a matched pair of parentheses.
     * If that is the case, just throw away the parentheses.
     */
    return eval(p + 1, q - 1);
  }
  else {
    op = the position of 主运算符 in the token expression;
    val1 = eval(p, op - 1);
    val2 = eval(op + 1, q);

    switch (op_type) {
      case '+': return val1 + val2;
      case '-': /* ... */
      case '*': /* ... */
      case '/': /* ... */
      default: assert(0);
    }
  }
}
```

详细代码太长不截图了, 这个功能所有相关代码位置:

nemu/src/isa/riscv32/reg.c

nemu/src/monitor/debug/expr.c

nemu/src/monitor/debug/ui.c

---

## 遇到的问题

### ▼ 问题1：运算错误，找不到主运算符

原因：几个条件语句一开始顺序反了。把判断加减乘除的放在最前面，导致左右括号无法正确识别使count改变。

解决后：

```
// get op
67 for(int i=p; i<=q; i++){
68     if(tokens[i].type=='('){
69         count++;
70         continue;
71     }
72     if(tokens[i].type==')'){
73         count--;
74         continue;
75     }
76     if(count!=0){
77         continue;
78     }
79     if(tokens[i].type!='+' && tokens[i].type!='-' && tokens[i].type!='*' &&
80 tokens[i].type!='/' && tokens[i].type!=TK_AND && tokens[i].type!=TK_OR){
81         continue;
82     }
83     int cur_pri = priority(tokens[i].type);
84     if(cur_pri >= op_pri){
85         op_pri = cur_pri;
86         op = i;
87         // printf("i=%d, op_pri = %d\n",i, cur_pri); //test
88     }
```

### ▼ 问题2：特殊情况处理

除数为零、主运算符找不到、单个输入有误等等特殊情况，打印一个ERROR信息方便调试。

---

## 5. 扫描内存

## 遇到的问题

### ▼ 问题1：address is out of bound

```
Welcome to riscv32-NEMU!  
For help, type "help"  
(nemu) x 4 5 *(2 +1)  
addr = 15  
address (0x0000000f) is out of bound {???} [0x00000000, 0x00000000] at pc = 0x80100000  
riscv32-nemu: src/device/io/map.c:20: check_bound: Assertion 'map != ((void *)0) && addr <= map->high && addr >= map->low' failed.  
make: *** [Makefile:77: run] 已放弃 (核心已转储)
```

原因：

- 记录物理内存的起始地址. 我们可以把内存看作一段连续的存储空间, 而内存又是字节编址的(即一个内存位置存放一个字节的的数据), 在C语言中我们就很自然地使用一个 `uint8_t` 类型的数组来对内存进行模拟.

NEMU默认为客户计算机提供128MB的物理内存

(见 `nemu/src/memory/memory.c` 中定义的 `pmem` ), 但对于一些ISA来说, 物

理内存并不是从0开始编址的, 例如mips32和riscv32的物理地址均

从 `0x80000000` 开始. 因此我们需要记录其物理内存的起始地址(通


过 `register_pmem()` 函数来完成), 将来CPU访问内存时, 我们会将CPU

将要访问的内存地址映射到 `pmem` 中的相应偏移位置. 例如如果mips32的

CPU打算访问内存地址 `0x80001234` , 我们最终会让它访

问 `pmem[0x1234]` . 这种机制有一个专门的名字, 叫地址映射, 在后续的PA

中我们还会再遇到它.

```
else  
    addr = addr + 0x80000000;   
    printf("addr = %08x\n", addr);  
    for(int i=0; i<N; i++){  
        printf("0x%08x: 0x%08x\n", addr+i*4, vaddr_read(addr+i*4, 4));  
    }
```

解决后：成功



```
WELCOME TO FUSEV2 NEMU!  
For help, type "help"  
(nemu) x 6 3*(2 +8)  
addr = 8000001e  
0x8000001e: 0x00000000  
0x80000022: 0x00000000  
0x80000026: 0x00000000  
0x8000002a: 0x00000000  
0x8000002e: 0x00000000  
0x80000032: 0x00000000  
(nemu)
```

## 6. 设置监视点

删除监视点

代码主要位置：

```
nemu/include/monitor/watchpoint.h  
nemu/src/monitor/debug/watchpoint.c  
nemu/src/monitor/debug/ui.c
```

```
10 void isa_reg_display() {  
11     int i;  
12  
13     /*Display register status*/  
14     for(i=0; i<32; i++){  
15         printf("%s =0x%08x\n", reg_name(i,0), reg_l(i));  
16     }  
17 }  
18
```



```
47 static int cmd_d(char *args);
48
49 static struct {
50     char *name;
51     char *description;
52     int (*handler) (char *);
53 } cmd_table [] = {
54     { "help", "Display informations about all supported commands", cmd_help },
55     { "c", "Continue the execution of the program", cmd_c },
56     { "q", "Exit NEMU", cmd_q },
57     { "si", "Stop the program after stepping N steps. N defaults to 1. (si [N])", cmd_si},
58     { "info", "Print program status.SUBCMD: r(register)/w(watchpoint). (info SUBCMD)", cmd_info},
59     { "p", "Expression evaluation. (p EXPR)", cmd_p},
60     { "x", "Scan memory. (x N EXPR)", cmd_x},
61     { "w", "Set Watchpoint. (w EXPR)", cmd_w},
62     { "d", "Cancel Watchpoint. (d N)", cmd_d},
63     /* TODO: Add more commands */
64 };
65
66
```

nemu/include/monitor/watchpoint.h

类型中加入表达式，和表达式值，然后通过enable判断是否在使用中

运行结果：

```

(nemu) w 100+2+31
Watchpoint set succeed.
(nemu) info w

```

NO	EXPR	VALUE
2	100+2+31	133
1	4+55	59
0	2+4	6

```

(nemu) d 1
Delete watchpoint.
(nemu) info w

```

NO	EXPR	VALUE
2	100+2+31	133
0	2+4	6

```

(nemu) w 10+10
Watchpoint set succeed.
(nemu) info w

```

NO	EXPR	VALUE
3	10+10	20
2	100+2+31	133
0	2+4	6

```

(nemu) w 10+10+10
Watchpoint set succeed.
(nemu) info w

```

NO	EXPR	VALUE
1	10+10+10	30
3	10+10	20
2	100+2+31	133
0	2+4	6

set

delete

序号回收