

Mark Lucernas  
CISC 211  
Prof. Saied Moezzi  
December 5, 2020

cisc-211-13@raspberrypi

## Unit 7 Assignment

### Overview Screen Capture

```
16 @ Assignment 6.14
17 @
18 @ Mark Lucernas
19 @
20 @ To run this code on a RaspberryPi (if the filename is filename.s) do the following
21 @ as -o filename.o filename.s
22 @ gcc -o filename filename.o
23 @ ./filename
24 @
25
26 7.text
27 8.global main
28 9.extern printf
29 10:main:
30 11:    @ push return address (lr) and fp on the stack
31 12:    push    {fp, lr}
32 13:
33 14:    @ R0 and R1 are the input, and they initially contain positive numbers, a and b (for example 9 and 2).
34 15:    @ At the end of the program, R0 is the output.
35 16:
36 17:    MOV R0, R0    @ move number 9 into register R0
37 18:    MOV R1, R2    @ move number 2 into register R1
39 19:
40 20:    @ Insert the ARM Assembly code for the following machine language:
41 21:    @ 0x00000000 0xE1A03000
42 22:    @ 0x00000010 0xE1510000
43 23:    @ 0x00000014 0xEA000002
44 24:    @ 0x00000018 0xE2822001
45 25:    @ 0x0000001C 0xE6811003
46 26:    @ 0x00000020 0xEAFFFFFA
47 27:    @ after the following two lines:
48 28:
49 29:
50 30:    MOV R2, #0    @ 1 = 0
51 31:    MOV R3, R1    @ R3 = R1 (divisor)
52 32:
53 33:    L1:
54 34:    CMP R1, R0    @ 1 < R0 (dividend)
55 35:    BHI DONE     @ if 1 >= R0 exit loop
56 36:    ADD R2, R2, #1 @ R2 = R2 + 1
57 37:    ADD R1, R1, R3 @ R1 = R2 + R3
58 38:    B L1         @ repeat loop
59 39:
60 40:    DONE:
61 41:    MOV R0, R2    @ R0 = R2
62 42:
63 43:    @ end of ARM Assembly code for the above machine language ends here.
64 44:
65 45:    @ the following code prints the content
66 46:
67 47:    MOV R1, R0
68 48:    LDR R0, =fmt   @ load fmt (output format)
69 49:    BL printf
70 50:
71 51:    @ pop the values from stack into fp and pc registers
72 52:    pop     {fp, pc}
73 53:
74 54:
75 55:
76 56:
77 57:
78 58:
79 59:
80 60:
81 61:
82 62:
83 63:
84 64:
85 65:
86 66:
87 67:
88 68:
89 69:
90 70:
91 71:
92 72:
93 73:
94 74:
95 75:
96 76:
97 77:
98 78:
99 79:
100 80:
101 81:
102 82:
103 83:
104 84:
105 85:
106 86:
107 87:
108 88:
109 89:
110 90:
111 91:
112 92:
113 93:
114 94:
115 95:
116 96:
117 97:
118 98:
119 99:
120 100:
121 101:
122 102:
123 103:
124 104:
125 105:
126 106:
127 107:
128 108:
129 109:
130 110:
131 111:
132 112:
133 113:
134 114:
135 115:
136 116:
137 117:
138 118:
139 119:
140 120:
141 121:
142 122:
143 123:
144 124:
145 125:
146 126:
147 127:
148 128:
149 129:
150 130:
151 131:
152 132:
153 133:
154 134:
155 135:
156 136:
157 137:
158 138:
159 139:
160 140:
161 141:
162 142:
163 143:
164 144:
165 145:
166 146:
167 147:
168 148:
169 149:
170 150:
171 151:
172 152:
173 153:
174 154:
175 155:
176 156:
177 157:
178 158:
179 159:
180 160:
181 161:
182 162:
183 163:
184 164:
185 165:
186 166:
187 167:
188 168:
189 169:
190 170:
191 171:
192 172:
193 173:
194 174:
195 175:
196 176:
197 177:
198 178:
199 179:
200 180:
201 181:
202 182:
203 183:
204 184:
205 185:
206 186:
207 187:
208 188:
209 189:
210 190:
211 191:
212 192:
213 193:
214 194:
215 195:
216 196:
217 197:
218 198:
219 199:
220 200:
221 201:
222 202:
223 203:
224 204:
225 205:
226 206:
227 207:
228 208:
229 209:
230 210:
231 211:
232 212:
233 213:
234 214:
235 215:
236 216:
237 217:
238 218:
239 219:
240 220:
241 221:
242 222:
243 223:
244 224:
245 225:
246 226:
247 227:
248 228:
249 229:
250 230:
251 231:
252 232:
253 233:
254 234:
255 235:
256 236:
257 237:
258 238:
259 239:
260 240:
261 241:
262 242:
263 243:
264 244:
265 245:
266 246:
267 247:
268 248:
269 249:
270 250:
271 251:
272 252:
273 253:
274 254:
275 255:
276 256:
277 257:
278 258:
279 259:
280 260:
281 261:
282 262:
283 263:
284 264:
285 265:
286 266:
287 267:
288 268:
289 269:
290 270:
291 271:
292 272:
293 273:
294 274:
295 275:
296 276:
297 277:
298 278:
299 279:
300 280:
301 281:
302 282:
303 283:
304 284:
305 285:
306 286:
307 287:
308 288:
309 289:
310 290:
311 291:
312 292:
313 293:
314 294:
315 295:
316 296:
317 297:
318 298:
319 299:
320 300:
321 301:
322 302:
323 303:
324 304:
325 305:
326 306:
327 307:
328 308:
329 309:
330 310:
331 311:
332 312:
333 313:
334 314:
335 315:
336 316:
337 317:
338 318:
339 319:
340 320:
341 321:
342 322:
343 323:
344 324:
345 325:
346 326:
347 327:
348 328:
349 329:
350 330:
351 331:
352 332:
353 333:
354 334:
355 335:
356 336:
357 337:
358 338:
359 339:
360 340:
361 341:
362 342:
363 343:
364 344:
365 345:
366 346:
367 347:
368 348:
369 349:
370 350:
371 351:
372 352:
373 353:
374 354:
375 355:
376 356:
377 357:
378 358:
379 359:
380 360:
381 361:
382 362:
383 363:
384 364:
385 365:
386 366:
387 367:
388 368:
389 369:
390 370:
391 371:
392 372:
393 373:
394 374:
395 375:
396 376:
397 377:
398 378:
399 379:
400 380:
401 381:
402 382:
403 383:
404 384:
405 385:
406 386:
407 387:
408 388:
409 389:
410 390:
411 391:
412 392:
413 393:
414 394:
415 395:
416 396:
417 397:
418 398:
419 399:
420 400:
421 401:
422 402:
423 403:
424 404:
425 405:
426 406:
427 407:
428 408:
429 409:
430 410:
431 411:
432 412:
433 413:
434 414:
435 415:
436 416:
437 417:
438 418:
439 419:
440 420:
441 421:
442 422:
443 423:
444 424:
445 425:
446 426:
447 427:
448 428:
449 429:
450 430:
451 431:
452 432:
453 433:
454 434:
455 435:
456 436:
457 437:
458 438:
459 439:
460 440:
461 441:
462 442:
463 443:
464 444:
465 445:
466 446:
467 447:
468 448:
469 449:
470 450:
471 451:
472 452:
473 453:
474 454:
475 455:
476 456:
477 457:
478 458:
479 459:
480 460:
481 461:
482 462:
483 463:
484 464:
485 465:
486 466:
487 467:
488 468:
489 469:
490 470:
491 471:
492 472:
493 473:
494 474:
495 475:
496 476:
497 477:
498 478:
499 479:
500 480:
501 481:
502 482:
503 483:
504 484:
505 485:
506 486:
507 487:
508 488:
509 489:
510 490:
511 491:
512 492:
513 493:
514 494:
515 495:
516 496:
517 497:
518 498:
519 499:
520 500:
521 501:
522 502:
523 503:
524 504:
525 505:
526 506:
527 507:
528 508:
529 509:
530 510:
531 511:
532 512:
533 513:
534 514:
535 515:
536 516:
537 517:
538 518:
539 519:
540 520:
541 521:
542 522:
543 523:
544 524:
545 525:
546 526:
547 527:
548 528:
549 529:
550 530:
551 531:
552 532:
553 533:
554 534:
555 535:
556 536:
557 537:
558 538:
559 539:
560 540:
561 541:
562 542:
563 543:
564 544:
565 545:
566 546:
567 547:
568 548:
569 549:
570 550:
571 551:
572 552:
573 553:
574 554:
575 555:
576 556:
577 557:
578 558:
579 559:
580 560:
581 561:
582 562:
583 563:
584 564:
585 565:
586 566:
587 567:
588 568:
589 569:
590 570:
591 571:
592 572:
593 573:
594 574:
595 575:
596 576:
597 577:
598 578:
599 579:
600 580:
601 581:
602 582:
603 583:
604 584:
605 585:
606 586:
607 587:
608 588:
609 589:
610 590:
611 591:
612 592:
613 593:
614 594:
615 595:
616 596:
617 597:
618 598:
619 599:
620 600:
621 601:
622 602:
623 603:
624 604:
625 605:
626 606:
627 607:
628 608:
629 609:
630 610:
631 611:
632 612:
633 613:
634 614:
635 615:
636 616:
637 617:
638 618:
639 619:
640 620:
641 621:
642 622:
643 623:
644 624:
645 625:
646 626:
647 627:
648 628:
649 629:
650 630:
651 631:
652 632:
653 633:
654 634:
655 635:
656 636:
657 637:
658 638:
659 639:
660 640:
661 641:
662 642:
663 643:
664 644:
665 645:
666 646:
667 647:
668 648:
669 649:
670 650:
671 651:
672 652:
673 653:
674 654:
675 655:
676 656:
677 657:
678 658:
679 659:
680 660:
681 661:
682 662:
683 663:
684 664:
685 665:
686 666:
687 667:
688 668:
689 669:
690 670:
691 671:
692 672:
693 673:
694 674:
695 675:
696 676:
697 677:
698 678:
699 679:
700 680:
701 681:
702 682:
703 683:
704 684:
705 685:
706 686:
707 687:
708 688:
709 689:
710 690:
711 691:
712 692:
713 693:
714 694:
715 695:
716 696:
717 697:
718 698:
719 699:
720 700:
721 701:
722 702:
723 703:
724 704:
725 705:
726 706:
727 707:
728 708:
729 709:
730 710:
731 711:
732 712:
733 713:
734 714:
735 715:
736 716:
737 717:
738 718:
739 719:
740 720:
741 721:
742 722:
743 723:
744 724:
745 725:
746 726:
747 727:
748 728:
749 729:
750 730:
751 731:
752 732:
753 733:
754 734:
755 735:
756 736:
757 737:
758 738:
759 739:
760 740:
761 741:
762 742:
763 743:
764 744:
765 745:
766 746:
767 747:
768 748:
769 749:
770 750:
771 751:
772 752:
773 753:
774 754:
775 755:
776 756:
777 757:
778 758:
779 759:
780 760:
781 761:
782 762:
783 763:
784 764:
785 765:
786 766:
787 767:
788 768:
789 769:
790 770:
791 771:
792 772:
793 773:
794 774:
795 775:
796 776:
797 777:
798 778:
799 779:
800 780:
801 781:
802 782:
803 783:
804 784:
805 785:
806 786:
807 787:
808 788:
809 789:
810 790:
811 791:
812 792:
813 793:
814 794:
815 795:
816 796:
817 797:
818 798:
819 799:
820 800:
821 801:
822 802:
823 803:
824 804:
825 805:
826 806:
827 807:
828 808:
829 809:
830 810:
831 811:
832 812:
833 813:
834 814:
835 815:
836 816:
837 817:
838 818:
839 819:
840 820:
841 821:
842 822:
843 823:
844 824:
845 825:
846 826:
847 827:
848 828:
849 829:
850 830:
851 831:
852 832:
853 833:
854 834:
855 835:
856 836:
857 837:
858 838:
859 839:
860 840:
861 841:
862 842:
863 843:
864 844:
865 845:
866 846:
867 847:
868 848:
869 849:
870 850:
871 851:
872 852:
873 853:
874 854:
875 855:
876 856:
877 857:
878 858:
879 859:
880 860:
881 861:
882 862:
883 863:
884 864:
885 865:
886 866:
887 867:
888 868:
889 869:
890 870:
891 871:
892 872:
893 873:
894 874:
895 875:
896 876:
897 877:
898 878:
899 879:
900 880:
901 881:
902 882:
903 883:
904 884:
905 885:
906 886:
907 887:
908 888:
909 889:
910 890:
911 891:
912 892:
913 893:
914 894:
915 895:
916 896:
917 897:
918 898:
919 899:
920 900:
921 901:
922 902:
923 903:
924 904:
925 905:
926 906:
927 907:
928 908:
929 909:
930 910:
931 911:
932 912:
933 913:
934 914:
935 915:
936 916:
937 917:
938 918:
939 919:
940 920:
941 921:
942 922:
943 923:
944 924:
945 925:
946 926:
947 927:
948 928:
949 929:
950 930:
951 931:
952 932:
953 933:
954 934:
955 935:
956 936:
957 937:
958 938:
959 939:
960 940:
961 941:
962 942:
963 943:
964 944:
965 945:
966 946:
967 947:
968 948:
969 949:
970 950:
971 951:
972 952:
973 953:
974 954:
975 955:
976 956:
977 957:
978 958:
979 959:
980 960:
981 961:
982 962:
983 963:
984 964:
985 965:
986 966:
987 967:
988 968:
989 969:
990 970:
991 971:
992 972:
993 973:
994 974:
995 975:
996 976:
997 977:
998 978:
999 979:
1000 980:
1001 981:
1002 982:
1003 983:
1004 984:
1005 985:
1006 986:
1007 987:
1008 988:
1009 989:
1010 990:
1011 991:
1012 992:
1013 993:
1014 994:
1015 995:
1016 996:
1017 997:
1018 998:
1019 999:
1020 1000:
1021 1001:
1022 1002:
1023 1003:
1024 1004:
1025 1005:
1026 1006:
1027 1007:
1028 1008:
1029 1009:
1030 1010:
1031 1011:
1032 1012:
1033 1013:
1034 1014:
1035 1015:
1036 1016:
1037 1017:
1038 1018:
1039 1019:
1040 1020:
1041 1021:
1042 1022:
1043 1023:
1044 1024:
1045 1025:
1046 1026:
1047 1027:
1048 1028:
1049 1029:
1050 1030:
1051 1031:
1052 1032:
1053 1033:
1054 1034:
1055 1035:
1056 1036:
1057 1037:
1058 1038:
1059 1039:
1060 1040:
1061 1041:
1062 1042:
1063 1043:
1064 1044:
1065 1045:
1066 1046:
1067 1047:
1068 1048:
1069 1049:
1070 1050:
1071 1051:
1072 1052:
1073 1053:
1074 1054:
1075 1055:
1076 1056:
1077 1057:
1078 1058:
1079 1059:
1080 1060:
1081 1061:
1082 1062:
1083 1063:
1084 1064:
1085 1065:
1086 1066:
1087 1067:
1088 1068:
1089 1069:
1090 1070:
1091 1071:
1092 1072:
1093 1073:
1094 1074:
1095 1075:
1096 1076:
1097 1077:
1098 1078:
1099 1079:
1100 1080:
1101 1081:
1102 1082:
1103 1083:
1104 1084:
1105 1085:
1106 1086:
1107 1087:
1108 1088:
1109 1089:
1110 1090:
1111 1091:
1112 1092:
1113 1093:
1114 1094:
1115 1095:
1116 1096:
1117 1097:
1118 1098:
1119 1099:
1120 1100:
1121 1101:
1122 1102:
1123 1103:
1124 1104:
1125 1105:
1126 1106:
1127 1107:
1128 1108:
1129 1109:
1130 1110:
1131 1111:
1132 1112:
1133 1113:
1134 1114:
1135 1115:
1136 1116:
1137 1117:
1138 1118:
1139 1119:
1140 1120:
1141 1121:
1142 1122:
1143 1123:
1144 1124:
1145 1125:
1146 1126:
1147 1127:
1148 1128:
1149 1129:
1150 1130:
1151 1131:
1152 1132:
1153 1133:
1154 1134:
1155 1135:
1156 1136:
1157 1137:
1158 1138:
1159 1139:
1160 1140:
1161 1141:
1162 1142:
1163 1143:
1164 1144:
1165 1145:
1166 1146:
1167 1147:
1168 1148:
1169 1149:
1170 1150:
1171 1151:
1172 1152:
1173 1153:
1174 1154:
1175 1155:
1176 1156:
1177 1157:
1178 1158:
1179 1159:
1180 1160:
1181 1161:
1182 1162:
1183 1163:
1184 1164:
1185 1165:
1186 1166:
1187 1167:
1188 1168:
1189 1169:
1190 1170:
1191 1171:
1192 1172:
1193 1173:
1194 1174:
1195 1175:
1196 1176:
1197 1177:
1198 1178:
1199 1179:
1200 1180:
1201 1181:
1202 1182:
1203 1183:
1204 1184:
1205 1185:
1206 1186:
1207 1187:
1208 1188:
1209 1189:
1210 1190:
1211 1191:
1212 1192:
1213 1193:
1214 1194:
1215 1195:
1216 1196:
1217 1197:
1218 1198:
1219 1199:
1220 1200:
1221 1201:
1222 1202:
1223 1203:
1224 1204:
1225 1205:
1226 1206:
1227 1207:
1228 1208:
1229 1209:
1230 1210:
1231 1211:
1232 1212:
1233 1213:
1234 1214:
1235 1215:
1236 1216:
1237 1217:
1238 1218:
1239 1219:
1240 1220:
1241 1221:
1242 1222:
1243 1223:
1244 1224:
1245 1225:
1246 1226:
1247 1227:
1248 1228:
1249 1229:
1250 1230:
1251 1231:
1252 1232:
1253 1233:
1254 1234:
1255 1235:
1256 1236:
1257 1237:
1258 1238:
1259 1239:
1260 1240:
1261 1241:
1262 1242:
1263 1243:
1264 1244:
1265 1245:
1266 1246:
1267 1247:
1268 1248:
1269 1249:
1270 1250:
1271 1251:
1272 1252:
1273 1253:
1274 1254:
1275 1255:
1276 1256:
1277 
```

## Assembly Source Code Screen Capture

```
16 @ Assignment 6.14
15 @
14 @ Mark Lucernas
13 @
12 @ To run this code on a RaspberryPi (if the filename is filename.s) do the following
11 @     as -o filename.o filename.s
10 @     gcc -o filename filename.o
9 @     ./filename
8
7 .text
6 .global main
5 .extern printf
4 main:
3     @ push return address (lr) and ip on the stack
2     push    {ip, lr}
1
0 |
1     @ R0 and R1 are the input, and they initially contain positive numbers, a and b (for example 9 and 2).
2     @ At the end of the program, R0 is the output.
3
4     MOV R0, #9      @ move number 9 into register R0
5     MOV R1, #2      @ move number 2 into register R1
6
7 @ Insert the ARM Assembly code for the following machine language:
8 @     0x00008008 0xE3A02000
9 @     0x0000800C 0xE1A03001
10 @     0x00008010 0xE1510000
11 @     0x00008014 0x8A000002
12 @     0x00008018 0xE2822001
13 @     0x0000801C 0xE0811003
14 @     0x00008020 0xEAFFFFFA
15 @ after the following two lines:
16 @-----
17
18     MOV R2, #0      @ i = 0
19     MOV R3, R1      @ R3 = R1 (divisor)
20 L1:
21     CMP R1, R0      @ i < R0 (dividend)
22     BHI DONE        @ if i >= R0 exit loop
23     ADD R2, R2, #1   @ R2 = R2 + 1
24     ADD R1, R1, R3   @ R1 = R2 + R3
25     B L1            @ repeat loop
26 DONE:
27     MOV R0, R2      @ R0 = R2
28
29 @ end of ARM Assembly code for the above machine language ends here.
30 @-----
31 @
32 @ the following code prints the content
33
34     MOV R1, R0
35     LDR R0, =fmt      @ load fmt (output format)
36     BL printf
37
38     @ pop the values from stack into ip and pc registers
39     pop     {ip, pc}
40
41 .data
42 fmt:     .asciz "\nThis code produces the following Result: %d\n"
43
44
```

### Edited Assembly Code Screen Capture

```
@ Insert the ARM Assembly code for the following machine language:
@      0x00008008 0xE3A02000
@      0x0000800C 0xE1A03001
@      0x00008010 0xE1510000
@      0x00008014 0x8A000002
@      0x00008018 0xE2822001
@      0x0000801C 0xE0811003
@      0x00008020 0xEAFFFFFA
@ after the following two lines:
@-----

        MOV R2, #0          @ i = 0
        MOV R3, R1          @ R3 = R1 (divisor)
L1:
        CMP R1, R0          @ i < R0 (dividend)
        BHI DONE            @ if i >= R0 exit loop
        ADD R2, R2, #1      @ R2 = R2 + 1
        ADD R1, R1, R3      @ R1 = R2 + R3
        B L1                @ repeat loop
DONE:
        MOV R0, R2          @ R0 = R2
```

### DDCA\_unit-7\_assignment.s

@ Assignment 6.14

@

@ Mark Lucernas

@

@ To run this code on a RaspberryPi (if the filename is filename.s) do the following

@ as -o filename.o filename.s

@ gcc -o filename filename.o

@ ./filename

.text

.global main

.extern printf

main:

@ push return address (lr) and ip on the stack  
push {ip, lr}

@ R0 and R1 are the input, and they initially contain positive numbers, a and b (for example 9 and 2).

@ At the end of the program, R0 is the output.

MOV R0, #9 @ move number 9 into register R0

MOV R1, #2 @ move number 2 into register R1

@ Insert the ARM Assembly code for the following machine language:

@ 0x00008008 0xE3A02000

@ 0x0000800C 0xE1A03001

@ 0x00008010 0xE1510000

@ 0x00008014 0x8A000002

@ 0x00008018 0xE2822001

@ 0x0000801C 0xE0811003

@ 0x00008020 0xEAFFFFFA

@ after the following two lines:

@-----

MOV R2, #0 @ i = 0

MOV R3, R1 @ R3 = R1 (divisor)

L1:

CMP R1, R0 @ i < R0 (dividend)

BHI DONE @ if i >= R0 exit loop

ADD R2, R2, #1 @ R2 = R2 + 1

ADD R1, R1, R3 @ R1 = R2 + R3

B L1 @ repeat loop

DONE:

MOV R0, R2 @ R0 = R2

@ end of ARM Assembly code for the above machine language ends here.

@-----

@

@ the following code prints the content

```
MOV R1, R0
LDR R0, =fmt @ load fmt (output format)
BL printf
```

@ pop the values from stack into ip and pc registers  
pop {ip, pc}

.data

fmt: .asciz "\nThis code produces the following Result: %d\n"

## GDB Debugger Screen Capture

```

+--Register group: general-----
r0      0x1      1
r2      0x7efff28c 2130702988
r4      0x104bc  66748
r6      0x1      1
r8      0x0      0
r10     0x76fff000 1996484608
r12     0x7efff1b0 2130702768
lr      0x76e79678 1994888824
cpsr    0x80000010 -2147483632
r1      0x7efff284 2130702980
r3      0x10440  66624
r5      0x2102c  135212
r7      0xa      10
r9      0x0      0
r11     0x0      0
sp      0x7efff128 0x7efff128
pc      0x10458 0x10458 <LOOP1+8>

0x10444 <main+4>    ldr    r5, [pc, #104] ; 0x104b4 <exit+4>
0x10448 <main+8>    mov    r6, #0
0x1044c <main+12>   mov    r7, #10
0x10450 <LOOP1>     cmp    r6, r7
0x10454 <LOOP1+4>   beq    0x10464 <END1>
> 0x10458 <LOOP1+8> str    r6, [r5, r6, lsl #2]
0x1045c <LOOP1+12>  add    r6, r6, #1
0x10460 <LOOP1+16>  b      0x10450 <LOOP1>
0x10464 <END1>     mov    r6, #0
0x10468 <LOOP2>     cmp    r6, r7
0x1046c <LOOP2+4>   beq    0x10488 <END2>
0x10470 <LOOP2+8>   lsl    r8, r6, #2
0x10474 <LOOP2+12>  ldr    r9, [r5, r8]
0x10478 <LOOP2+16>  add    r9, r9, #10

native process 8960 In: LOOP1
(gdb) stepi
0x00010448 in main ()
0x0001044c in main ()
0x00010450 in LOOP1 ()
0x00010454 in LOOP1 ()
0x00010458 in LOOP1 ()
0x0001045c in LOOP1 ()
0x00010460 in LOOP1 ()
0x00010450 in LOOP1 ()
0x00010454 in LOOP1 ()
0x00010458 in LOOP1 ()
(gdb)
```

## C++ Source Code Screen Capture

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{
    int dividend = 9;           // R0 = A (dividend)
    int divisor = 2;           // R1 = B (divisor)
    int i = 0;                 // MOV R2, #0
    int tmp = divisor;         // MOV R3, R1

    int quotient;

    // L1
    while (dividend >= tmp) {   // CMP R1, R0 ? Proceed : BHI DONE
        i = i + 1;             // ADD R2, R2, #1
        tmp = tmp + divisor;    // ADD R1, R1, R3
    }

    // DONE
    quotient = i;              // MOV R0, R2

    cout << "\nThis code produces the following Result: " << quotient << endl;
    return 0;
}
```

### *DDCA\_unit-7.cpp*

```
#include <iostream>
using namespace std;
```

```
int main(int argc, char *argv[])
{
```

```
    int dividend = 9;           // R0 = A (dividend)
    int divisor = 2;           // R1 = B (divisor)
    int i = 0;                 // MOV R2, #0
    int tmp = divisor;         // MOV R3, R1
```

```
    int quotient;
```

```
    // L1
```

```
    while (dividend >= tmp) {   // CMP R1, R0 ? Proceed : BHI DONE
        i = i + 1;             // ADD R2, R2, #1
        tmp = tmp + divisor;    // ADD R1, R1, R3
    }
```

```
// DONE
quotient = i;          // MOV R0, R2

cout << "\nThis code produces the following Result: " << quotient << endl;
return 0;
}
}
```