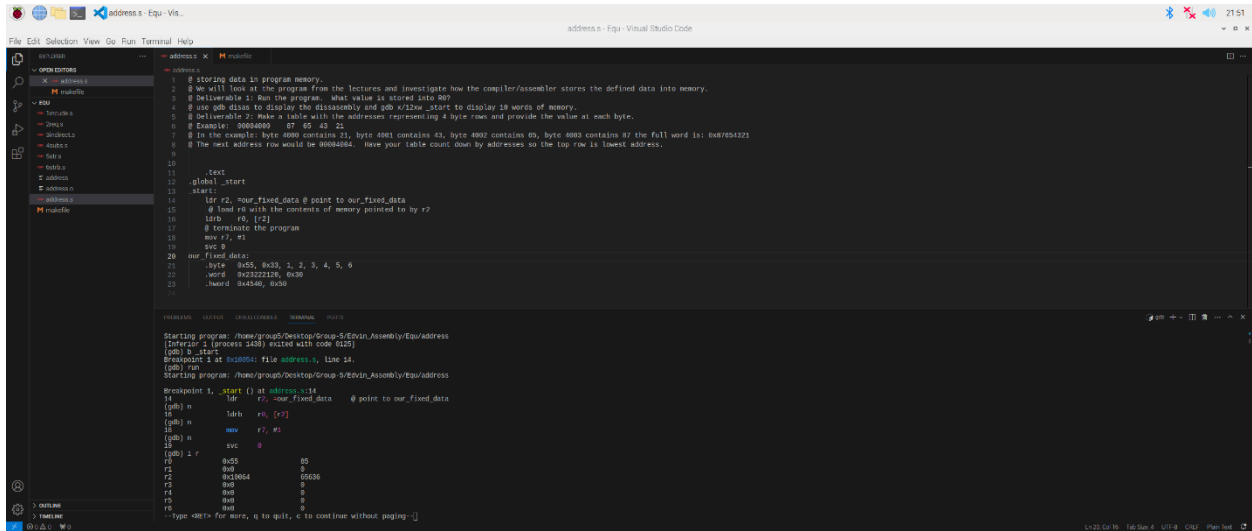


1. 0_AddressMap

a. The value stored in R0 after the program has been run is 85 in decimal.

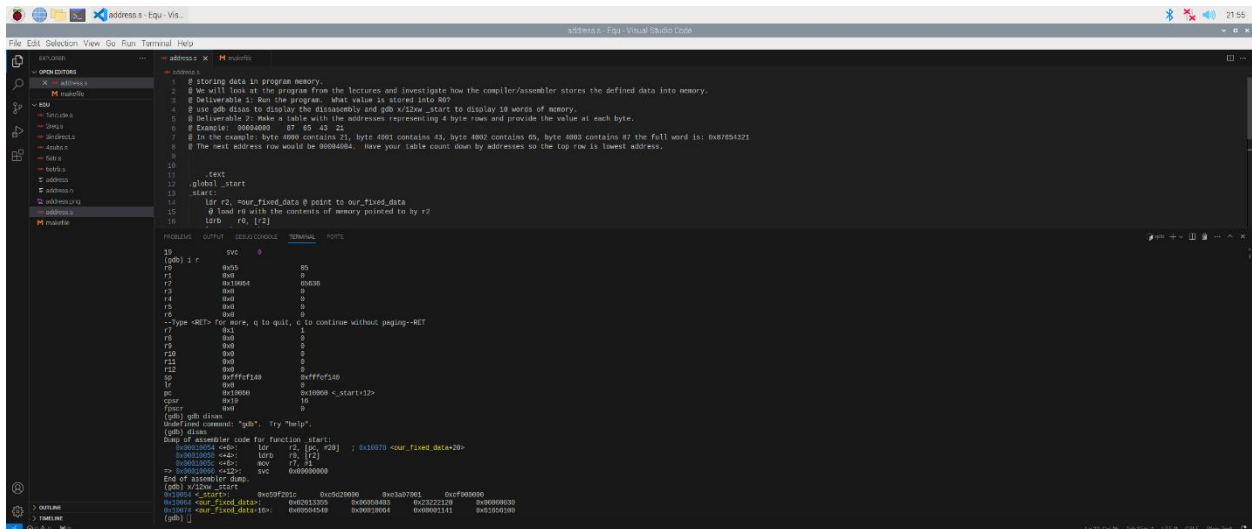


```
1 # storing data in program memory.
2 # we will look at the program from the lectures and investigate how the compiler/assembler stores the defined data into memory.
3 # Deliverable 1: Run the program. What value is stored into R0?
4 # use gdb disas to display the disassembly and gdb x/12w _start to display 12 words of memory.
5 # Deliverable 2: Make a table with the addresses representing 4 byte rows and provide the value at each byte.
6 # Example: 00000000 - 07 05 43 75
7 # In the example: byte 0000 contains 21, byte 0001 contains 43, byte 0002 contains 65, byte 0003 contains 87 the full word is: 0x07054375
8 # The next address row would be 00000004. Have your table count down by addresses so the top row is lowest address.
9
10
11 .text
12 .global _start
13 _start:
14     ldr r2, <our_fixed_data> @ point to our_fixed_data
15     @ load r0 with the contents of memory pointed to by r2
16     ldrb r0, [r2]
17     @ terminate the program
18     mov r7, #1
19     svc #0
20
21 our_fixed_data:
22     .byte 0x55, 0x33, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6
23     .word 0x3222123, 0x30
24     .word 0x4567, 0x56
```

Starting program: /home/group/Desktop/group-5/0b/in_Asembly/Equ/address
[inferior 1 (process 1438) exited with code 0125]
[gdb] 3 _start
Breakpoint 1 at 0x10054: file address.s, line 14.
[gdb] run
Starting program: /home/group/Desktop/group-5/0b/in_Asembly/Equ/address
Breakpoint 1, _start () at address.s:14
14 ldr r2, <our_fixed_data> @ point to our_fixed_data
[gdb] n
16 ldrb r0, [r2]
[gdb] n
18 mov r7, #1
[gdb] n
19 svc #0
[gdb] i r
r0 0x55 85
r1 0x0 0
r2 0x10054 0x10054
r3 0x0 0
r4 0x0 0
r5 0x0 0
r6 0x0 0
r7 0x1 1
r8 0x0 0
r9 0x0 0
r10 0x0 0
r11 0x0 0
r12 0x0 0
r13 0x0 0
r14 0x0 0
r15 0x0 0
--type <RET> for more, q to quit, c to continue without paging--

b.

4000	00	01	00	54
4004	00	01	00	58
4008	00	01	00	5c
4012	00	01	00	00



```
1 # storing data in program memory.
2 # we will look at the program from the lectures and investigate how the compiler/assembler stores the defined data into memory.
3 # Deliverable 1: Run the program. What value is stored into R0?
4 # use gdb disas to display the disassembly and gdb x/12w _start to display 12 words of memory.
5 # Deliverable 2: Make a table with the addresses representing 4 byte rows and provide the value at each byte.
6 # Example: 00000000 - 07 05 43 75
7 # In the example: byte 0000 contains 21, byte 0001 contains 43, byte 0002 contains 65, byte 0003 contains 87 the full word is: 0x07054375
8 # The next address row would be 00000004. Have your table count down by addresses so the top row is lowest address.
9
10
11 .text
12 .global _start
13 _start:
14     ldr r2, <our_fixed_data> @ point to our_fixed_data
15     @ load r0 with the contents of memory pointed to by r2
16     ldrb r0, [r2]
17     @ terminate the program
18     mov r7, #1
19     svc #0
20
21 our_fixed_data:
22     .byte 0x55, 0x33, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6
23     .word 0x3222123, 0x30
24     .word 0x4567, 0x56
```

Starting program: /home/group/Desktop/group-5/0b/in_Asembly/Equ/address
[inferior 1 (process 1438) exited with code 0125]
[gdb] 3 _start
Breakpoint 1 at 0x10054: file address.s, line 14.
[gdb] run
Starting program: /home/group/Desktop/group-5/0b/in_Asembly/Equ/address
Breakpoint 1, _start () at address.s:14
14 ldr r2, <our_fixed_data> @ point to our_fixed_data
[gdb] n
16 ldrb r0, [r2]
[gdb] n
18 mov r7, #1
[gdb] n
19 svc #0
[gdb] i r
r0 0x55 85
r1 0x0 0
r2 0x10054 0x10054
r3 0x0 0
r4 0x0 0
r5 0x0 0
r6 0x0 0
r7 0x1 1
r8 0x0 0
r9 0x0 0
r10 0x0 0
r11 0x0 0
r12 0x0 0
r13 0x0 0
r14 0x0 0
r15 0x0 0
--type <RET> for more, q to quit, c to continue without paging--RET
[gdb] q
[gdb] disas
Dump of assembler code for function _start:
0x10054000 <+>: ldr r2, [r2] ; 0x10054000 <our_fixed_data>
0x10054004 <+>: ldrb r0, [r2]
0x10054008 <+>: mov r7, #1
0x1005400c <+>: svc #0
End of assembler dump.
[gdb] x/12w _start
0x10054000: 0x10054000 0x10054000 0x10054000 0x10054000
0x10054004: 0x10054004 0x10054004 0x10054004 0x10054004
0x10054008: 0x10054008 0x10054008 0x10054008 0x10054008
0x1005400c: 0x1005400c 0x1005400c 0x1005400c 0x1005400c
[gdb]]

2. 1_AddInclude

a.

```

1 //Look at the unit1.s file. This introduces the .req directive which assigns values to variable.
2 //Deliverable 1: Compile and run the program
3 //Deliverable 2: Change the last two lines to use meaningful variable names from the classinclude.s file. Run your program.
4 //Deliverable 3: What does the .include mean/do in the program?
5
6 .include "classinclude.s"
7
8 .global start
9
10 .start:
11
12 //This stores 20 in R4 then subtracts 10 from that and stores the remaining amount in R0
13 MOV R4, #20
14 ADD R4, R4, #R10
15 MOV R7, R4
16 SMT R
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
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

```

b.

```

1 //Look at the unit1.s file. This introduces the .req directive which assigns values to variable.
2 //Deliverable 1: Compile and run the program
3 //Deliverable 2: Change the last two lines to use meaningful variable names from the classinclude.s file. Run your program.
4 //Deliverable 3: What does the .include mean/do in the program?
5
6 .include "classinclude.s"
7
8 .global start
9
10 .start:
11
12 //This stores 20 in R4 then subtracts 10 from that and stores the remaining amount in R0
13 MOV R4, #20
14 ADD R4, R4, #R10
15 MOV R7, R4
16 SMT R
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
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

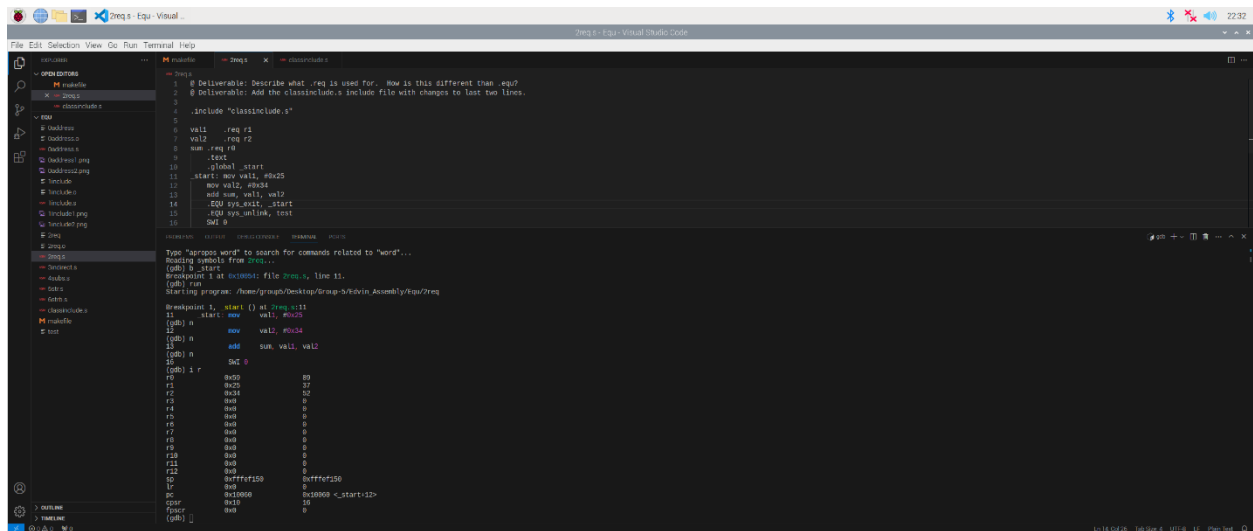
```

c. The include function allows another file to be accessed by the current program. This increases the modularity of programs, aiding in making programs easier to customize to specific roles.

3. 2_req_p2_2

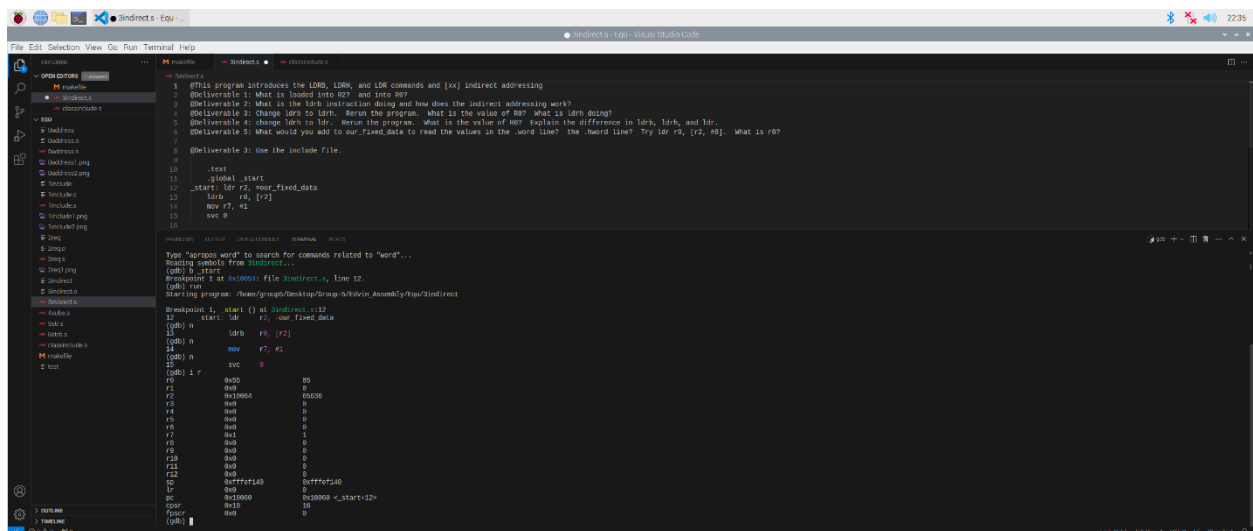
a. The role of the .req function is to assign a name to a register for the purpose of better clarity when writing scripts.

b.

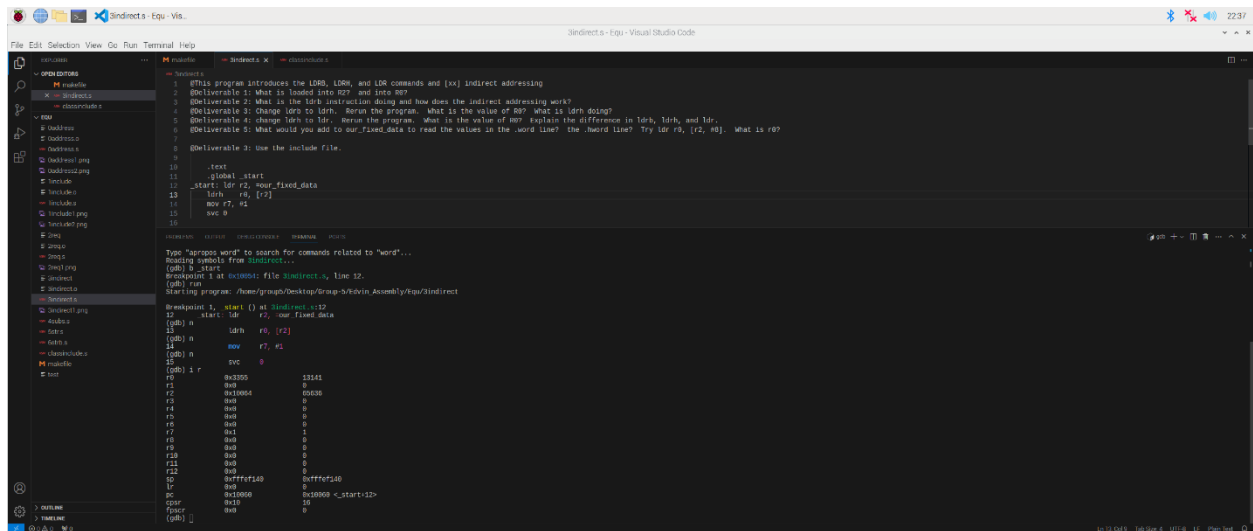


4. 3_Indirect2_3a

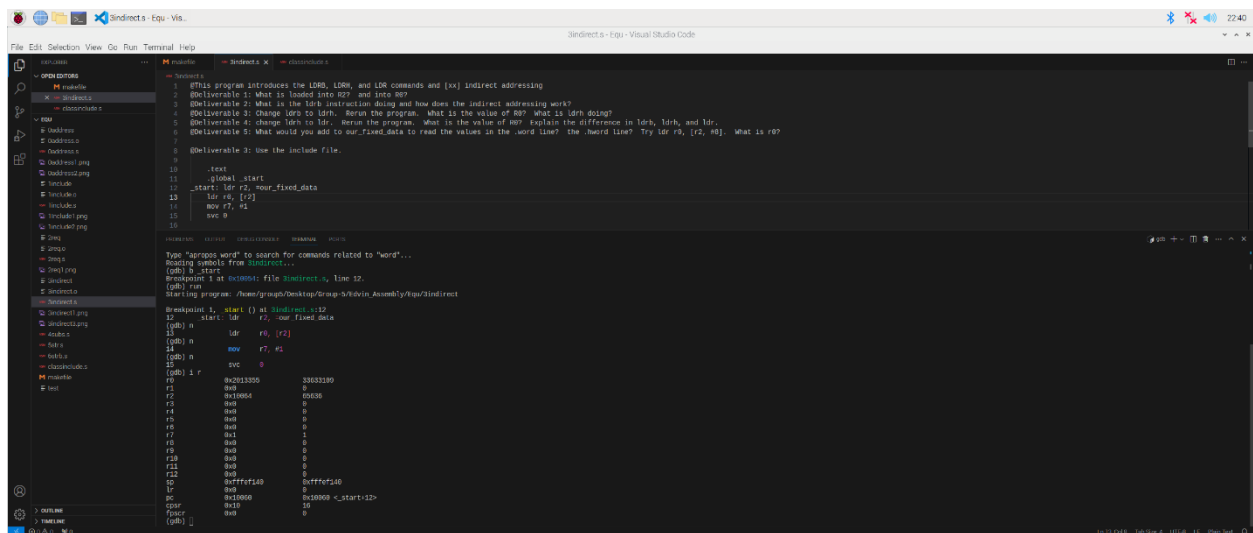
- 65,636 in decimal is loaded into R2. 85 in decimal is loaded into R0



- The LDRB is used to load a single byte of data from the memory onto a register. Indirect addressing works by first loading the data to a register and reading from there instead of directly from the memory.
- 13,141 in decimal is loaded into R0. This command loads the half-words in R2, which are 0x33 and 0x55 in that order.

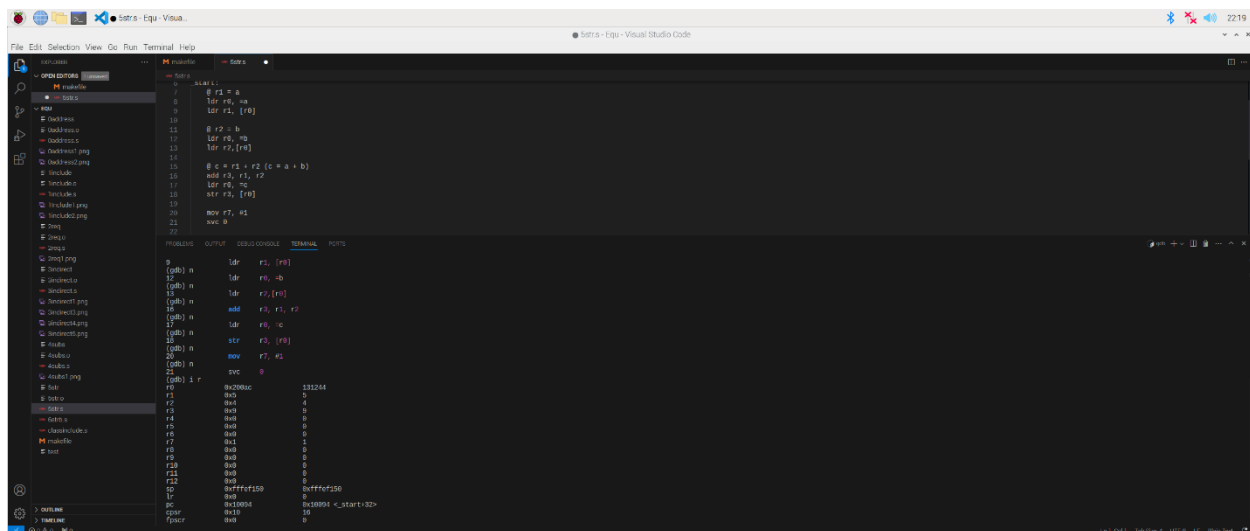


- d. 33633100 in decimal is loaded to R0. The difference between ldrb, ldrh, and ldr is that ldrb loads a single byte, ldrh loads a half-word, and ldr loads a full register.



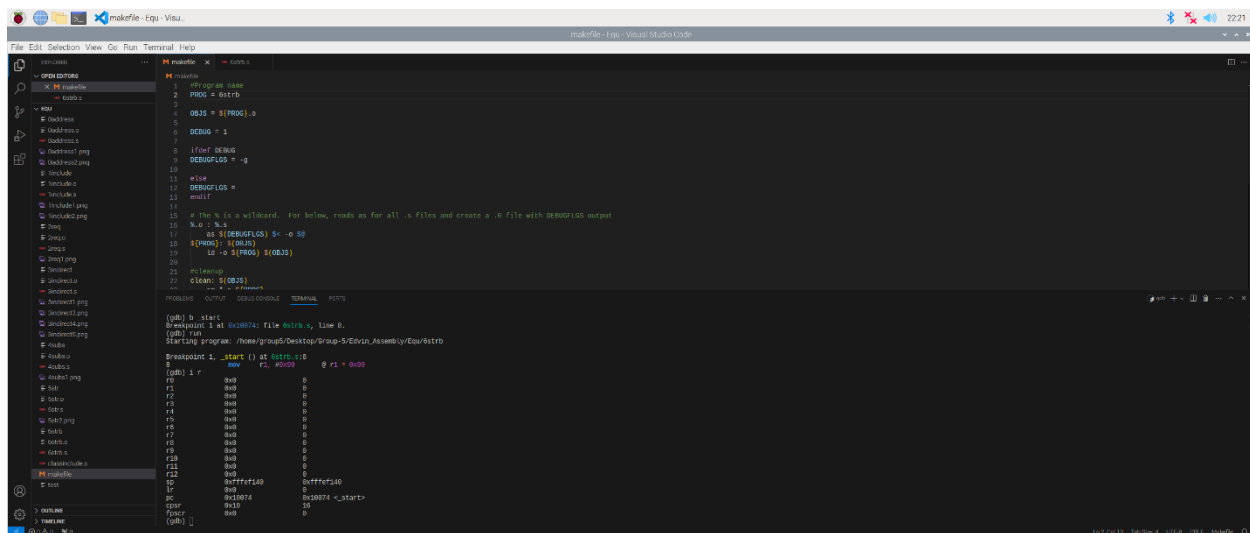
- e. You can specify the ldr function to skip a specified number of bytes to access other lines of data.

- The str instruction saves the data on a register to the memory.
-



7. 6_strb_ex2_3

a.



The screenshot shows the Emona IDE with the following assembly code in the main editor:

```

12: add r6, r6, r2 @ r6 = r6 + 1
13: mov r1, #0x05 @ r1 = 0x05
14: strb r1, [r6] @ store r1 into location pointed to by r6
15: @
16: add r6, r6, r1 @ r6 = r6 + 1
17: mov r1, #0x12 @ r1 = 0x12
18: strb r1, [r6] @ store r1 into location pointed to by r6
19: @
20: add r6, r6, r2 @ r6 = r6 + 1
21: mov r1, #0x03 @ r1 = 0x03
22: strb r1, [r6] @ store r1 into location pointed to by r6
23: @
24: add r6, r6, r1 @ r6 = r6 + 1
25: mov r1, #0x12 @ r1 = 0x12
26: strb r1, [r6]
27:
28: mov r7, r1
29: wch 0
30:
31: data .data
32: data_store: .space 8
33:
34:
35:

```

The register dump at the bottom shows the following values:

Register	Value	Comment
r0	0x00	
r1	0x05	
r2	0x00	
r3	0x00	
r4	0x00	
r5	0x00	
r6	0x0000	
r7	0x00	
r8	0x00	
r9	0x00	
r10	0x00	
r11	0x00	
r12	0x00	
sp	0xfffff140	
pc	0x1000	
pscr	0x10	
cpsr	0x00	

- The strb function stores the least significant byte in the register to the memory.
- This reserves a block of memory space without initializing it.