

6510 +

Unlock the true programming power of your C64 with this excellent machine language assembler.

As assembler for all those who haven't been following the Power Packbooks, is a program that takes your assembly language

instructions and turns them into a machine code program that your C64's processor chip can understand. BASIC, essentially, is just a machine code program, you see, so BASIC programs have got to be run by a program that's taking care of the chip — like programs do the job of cars, to reflect. Take it easy. Programming in machine code means you're running the program directly, making use of the machine's own statistics, making that program faster and more powerful — if your machine can do it, you can make it do it in machine code.

Now there you don't need this? (Every article here that's ever been given a spot) well, how have we been in machine code — BASIC just can't cut it. You still need to know how to use this program. Though, don't you?

Right — the program you'll find on this disk will install a copy of 6510+ on the tape or disk of your choice. The package that'll say this computer to go into here, but anyone who's been coding for any amount of time will already know exactly how to do it.

What if you're a beginner? What I can't explain here is program in machine code that

isn't a new feature starting on page 33 that shows all that stuff, but can explain how the assembler works, so you'll be able to use it with the various excellent technical features OFFER.

Your listing is written in the form of a BASIC program using comments as opposed to normal BASIC comments. But 6510+ will take and assemble, in other words, when you save each line of the machine language listing you want to assemble and give it a line number

in the normal BASIC way after the word **assembly** and number that's the list that doesn't tell you what the code is. (Normally, you can place a comment which will act like a BASIC statement, though it will have to be preceded with a ":", or a "I".



ASSEMBLY DIRECTIVES

that all comments have to be comments. In the way — there are some directives that, instead of assembling into real instructions, give the assembler (that specific instructions) if you're open to assembly language most of what makes machine code, but it will save if you carry on reading the disk Machine manual.

6510+ — This is used to place bytes directly into memory. This cannot be used for this command would be something like `xxx, 11, 11, 11`, which would place the bytes 11, 11 and 11 consecutively. By the same token, you can give the command a list of bytes and the assembler will place the 6510 values of each character into memory (memory).

1000 — This is the same as `xxx`, but you can use it if you're used to that command instead (some people prefer it that way apparently).

1001 — This one places a low-byte word into memory. This command is used to indicate the use of words.

1002 — This can store one `xxx`, or four.

Comments are used to clarify how source files together. If you need one, "Comment", the

assembler would switch to the new source file to continue assembly. Bear in mind that all changed files must end with an asterisk. (L20) — This is very similar to `xxx`, but allows you to insert a chunk of source code from elsewhere into your listing. This command doesn't do much unless files are made another one to.

ASSEMBLE IT, THEN

When you've finished your program, you'll want to assemble it, won't you. The easy way to do this is with the **ASSEMBLE** command, that doesn't need any other parameters. If you'd like to store assembly from a particular line number, though, you can state that the number (**ASSEMBLE 100** and **6510** will happily oblige).

One point — 6510+ handles a few other BASIC commands in order that it can work properly. While this is great and everything, it does mean that some normal BASIC programs may not work with the assembler to memory. This has been noted — other uses of running normal BASIC programs with it.



MAYHEM MEGAMIX

More melodic madness from the master of chaos, John and Steve Rosebush, for each and every one of the greatest games made in the history of gaming, all of 'em from Space Invaders, Bowling Greenhouse, Christmas Island, Mayhem in Wonderland. And it's all wrapped up in one of the best-looking action systems you've ever

seen. Just hit the function keys to select what game you want next, then hit the enter key corresponding to the item that you want to load. To start and stop the music, give it a few other effects.

Use the Mayhem cursor at the bottom of the screen, which is controlled by your joystick. Then all that's left.



QUICKSTART INFO

- PLAY** Select game.
- END** Select item.
- ENTER** Start Mayhem.

Other info You select the various functions (L20) also, with the Mayhem Megamix app, the instructions at the bottom of the screen and pressing the `xxx` button, it's got to be one of the best-looking action systems you've ever seen.

6510+ IDIOT'S GUIDE

We're going to use a program from Steve Rosebush, the manual notes that state this number on page 10. The listing can be found under the heading "How to use" within the manual, and there's how to use it with 6510+.

- 1 Load 6510+.
- 2 Enter the following program in BASIC:
`10 GOTO 1000`
`1000 GOTO 1001`
`1001 GOTO 1002`
`1002 GOTO 1003`
`1003 GOTO 1004`
`1004 GOTO 1005`
`1005 GOTO 1006`
`1006 GOTO 1007`
`1007 GOTO 1008`
`1008 GOTO 1009`
`1009 GOTO 1010`
`1010 GOTO 1011`
`1011 GOTO 1012`
`1012 GOTO 1013`
`1013 GOTO 1014`
`1014 GOTO 1015`
`1015 GOTO 1016`
`1016 GOTO 1017`
`1017 GOTO 1018`
`1018 GOTO 1019`
`1019 GOTO 1020`
`1020 GOTO 1021`
`1021 GOTO 1022`
`1022 GOTO 1023`
`1023 GOTO 1024`
`1024 GOTO 1025`
`1025 GOTO 1026`
`1026 GOTO 1027`
`1027 GOTO 1028`
`1028 GOTO 1029`
`1029 GOTO 1030`
`1030 GOTO 1031`
`1031 GOTO 1032`
`1032 GOTO 1033`
`1033 GOTO 1034`
`1034 GOTO 1035`
`1035 GOTO 1036`
`1036 GOTO 1037`
`1037 GOTO 1038`
`1038 GOTO 1039`
`1039 GOTO 1040`
`1040 GOTO 1041`
`1041 GOTO 1042`
`1042 GOTO 1043`
`1043 GOTO 1044`
`1044 GOTO 1045`
`1045 GOTO 1046`
`1046 GOTO 1047`
`1047 GOTO 1048`
`1048 GOTO 1049`
`1049 GOTO 1050`
`1050 GOTO 1051`
`1051 GOTO 1052`
`1052 GOTO 1053`
`1053 GOTO 1054`
`1054 GOTO 1055`
`1055 GOTO 1056`
`1056 GOTO 1057`
`1057 GOTO 1058`
`1058 GOTO 1059`
`1059 GOTO 1060`
`1060 GOTO 1061`
`1061 GOTO 1062`
`1062 GOTO 1063`
`1063 GOTO 1064`
`1064 GOTO 1065`
`1065 GOTO 1066`
`1066 GOTO 1067`
`1067 GOTO 1068`
`1068 GOTO 1069`
`1069 GOTO 1070`
`1070 GOTO 1071`
`1071 GOTO 1072`
`1072 GOTO 1073`
`1073 GOTO 1074`
`1074 GOTO 1075`
`1075 GOTO 1076`
`1076 GOTO 1077`
`1077 GOTO 1078`
`1078 GOTO 1079`
`1079 GOTO 1080`
`1080 GOTO 1081`
`1081 GOTO 1082`
`1082 GOTO 1083`
`1083 GOTO 1084`
`1084 GOTO 1085`
`1085 GOTO 1086`
`1086 GOTO 1087`
`1087 GOTO 1088`
`1088 GOTO 1089`
`1089 GOTO 1090`
`1090 GOTO 1091`
`1091 GOTO 1092`
`1092 GOTO 1093`
`1093 GOTO 1094`
`1094 GOTO 1095`
`1095 GOTO 1096`
`1096 GOTO 1097`
`1097 GOTO 1098`
`1098 GOTO 1099`
`1099 GOTO 1100`
`1100 GOTO 1101`
`1101 GOTO 1102`
`1102 GOTO 1103`
`1103 GOTO 1104`
`1104 GOTO 1105`
`1105 GOTO 1106`
`1106 GOTO 1107`
`1107 GOTO 1108`
`1108 GOTO 1109`
`1109 GOTO 1110`
`1110 GOTO 1111`
`1111 GOTO 1112`
`1112 GOTO 1113`
`1113 GOTO 1114`
`1114 GOTO 1115`
`1115 GOTO 1116`
`1116 GOTO 1117`
`1117 GOTO 1118`
`1118 GOTO 1119`
`1119 GOTO 1120`
`1120 GOTO 1121`
`1121 GOTO 1122`
`1122 GOTO 1123`
`1123 GOTO 1124`
`1124 GOTO 1125`
`1125 GOTO 1126`
`1126 GOTO 1127`
`1127 GOTO 1128`
`1128 GOTO 1129`
`1129 GOTO 1130`
`1130 GOTO 1131`
`1131 GOTO 1132`
`1132 GOTO 1133`
`1133 GOTO 1134`
`1134 GOTO 1135`
`1135 GOTO 1136`
`1136 GOTO 1137`
`1137 GOTO 1138`
`1138 GOTO 1139`
`1139 GOTO 1140`
`1140 GOTO 1141`
`1141 GOTO 1142`
`1142 GOTO 1143`
`1143 GOTO 1144`
`1144 GOTO 1145`
`1145 GOTO 1146`
`1146 GOTO 1147`
`1147 GOTO 1148`
`1148 GOTO 1149`
`1149 GOTO 1150`
`1150 GOTO 1151`
`1151 GOTO 1152`
`1152 GOTO 1153`
`1153 GOTO 1154`
`1154 GOTO 1155`
`1155 GOTO 1156`
`1156 GOTO 1157`
`1157 GOTO 1158`
`1158 GOTO 1159`
`1159 GOTO 1160`
`1160 GOTO 1161`
`1161 GOTO 1162`
`1162 GOTO 1163`
`1163 GOTO 1164`
`1164 GOTO 1165`
`1165 GOTO 1166`
`1166 GOTO 1167`
`1167 GOTO 1168`
`1168 GOTO 1169`
`1169 GOTO 1170`
`1170 GOTO 1171`
`1171 GOTO 1172`
`1172 GOTO 1173`
`1173 GOTO 1174`
`1174 GOTO 1175`
`1175 GOTO 1176`
`1176 GOTO 1177`
`1177 GOTO 1178`
`1178 GOTO 1179`
`1179 GOTO 1180`
`1180 GOTO 1181`
`1181 GOTO 1182`
`1182 GOTO 1183`
`1183 GOTO 1184`
`1184 GOTO 1185`
`1185 GOTO 1186`
`1186 GOTO 1187`
`1187 GOTO 1188`
`1188 GOTO 1189`
`1189 GOTO 1190`
`1190 GOTO 1191`
`1191 GOTO 1192`
`1192 GOTO 1193`
`1193 GOTO 1194`
`1194 GOTO 1195`
`1195 GOTO 1196`
`1196 GOTO 1197`
`1197 GOTO 1198`
`1198 GOTO 1199`
`1199 GOTO 1200`
`1200 GOTO 1201`
`1201 GOTO 1202`
`1202 GOTO 1203`
`1203 GOTO 1204`
`1204 GOTO 1205`
`1205 GOTO 1206`
`1206 GOTO 1207`
`1207 GOTO 1208`
`1208 GOTO 1209`
`1209 GOTO 1210`
`1210 GOTO 1211`
`1211 GOTO 1212`
`1212 GOTO 1213`
`1213 GOTO 1214`
`1214 GOTO 1215`
`1215 GOTO 1216`
`1216 GOTO 1217`
`1217 GOTO 1218`
`1218 GOTO 1219`
`1219 GOTO 1220`
`1220 GOTO 1221`
`1221 GOTO 1222`
`1222 GOTO 1223`
`1223 GOTO 1224`
`1224 GOTO 1225`
`1225 GOTO 1226`
`1226 GOTO 1227`
`1227 GOTO 1228`
`1228 GOTO 1229`
`1229 GOTO 1230`
`1230 GOTO 1231`
`1231 GOTO 1232`
`1232 GOTO 1233`
`1233 GOTO 1234`
`1234 GOTO 1235`
`1235 GOTO 1236`
`1236 GOTO 1237`
`1237 GOTO 1238`
`1238 GOTO 1239`
`1239 GOTO 1240`
`1240 GOTO 1241`
`1241 GOTO 1242`
`1242 GOTO 1243`
`1243 GOTO 1244`
`1244 GOTO 1245`
`1245 GOTO 1246`
`1246 GOTO 1247`
`1247 GOTO 1248`
`1248 GOTO 1249`
`1249 GOTO 1250`
`1250 GOTO 1251`
`1251 GOTO 1252`
`1252 GOTO 1253`
`1253 GOTO 1254`
`1254 GOTO 1255`
`1255 GOTO 1256`
`1256 GOTO 1257`
`1257 GOTO 1258`
`1258 GOTO 1259`
`1259 GOTO 1260`
`1260 GOTO 1261`
`1261 GOTO 1262`
`1262 GOTO 1263`
`1263 GOTO 1264`
`1264 GOTO 1265`
`1265 GOTO 1266`
`1266 GOTO 1267`
`1267 GOTO 1268`
`1268 GOTO 1269`
`1269 GOTO 1270`
`1270 GOTO 1271`
`1271 GOTO 1272`
`1272 GOTO 1273`
`1273 GOTO 1274`
`1274 GOTO 1275`
`1275 GOTO 1276`
`1276 GOTO 1277`
`1277 GOTO 1278`
`1278 GOTO 1279`
`1279 GOTO 1280`
`1280 GOTO 1281`
`1281 GOTO 1282`
`1282 GOTO 1283`
`1283 GOTO 1284`
`1284 GOTO 1285`
`1285 GOTO 1286`
`1286 GOTO 1287`
`1287 GOTO 1288`
`1288 GOTO 1289`
`1289 GOTO 1290`
`1290 GOTO 1291`
`1291 GOTO 1292`
`1292 GOTO 1293`
`1293 GOTO 1294`
`1294 GOTO 1295`
`1295 GOTO 1296`
`1296 GOTO 1297`
`1297 GOTO 1298`
`1298 GOTO 1299`
`1299 GOTO 1300`
`1300 GOTO 1301`
`1301 GOTO 1302`
`1302 GOTO 1303`
`1303 GOTO 1304`
`1304 GOTO 1305`
`1305 GOTO 1306`
`1306 GOTO 1307`
`1307 GOTO 1308`
`1308 GOTO 1309`
`1309 GOTO 1310`
`1310 GOTO 1311`
`1311 GOTO 1312`
`1312 GOTO 1313`
`1313 GOTO 1314`
`1314 GOTO 1315`
`1315 GOTO 1316`
`1316 GOTO 1317`
`1317 GOTO 1318`
`1318 GOTO 1319`
`1319 GOTO 1320`
`1320 GOTO 1321`
`1321 GOTO 1322`
`1322 GOTO 1323`
`1323 GOTO 1324`
`1324 GOTO 1325`
`1325 GOTO 1326`
`1326 GOTO 1327`
`1327 GOTO 1328`
`1328 GOTO 1329`
`1329 GOTO 1330`
`1330 GOTO 1331`
`1331 GOTO 1332`
`1332 GOTO 1333`
`1333 GOTO 1334`
`1334 GOTO 1335`
`1335 GOTO 1336`
`1336 GOTO 1337`
`1337 GOTO 1338`
`1338 GOTO 1339`
`1339 GOTO 1340`
`1340 GOTO 1341`
`1341 GOTO 1342`
`1342 GOTO 1343`
`1343 GOTO 1344`
`1344 GOTO 1345`
`1345 GOTO 1346`
`1346 GOTO 1347`
`1347 GOTO 1348`
`1348 GOTO 1349`
`1349 GOTO 1350`
`1350 GOTO 1351`
`1351 GOTO 1352`
`1352 GOTO 1353`
`1353 GOTO 1354`
`1354 GOTO 1355`
`1355 GOTO 1356`
`1356 GOTO 1357`
`1357 GOTO 1358`
`1358 GOTO 1359`
`1359 GOTO 1360`
`1360 GOTO 1361`
`1361 GOTO 1362`
`1362 GOTO 1363`
`1363 GOTO 1364`
`1364 GOTO 1365`
`1365 GOTO 1366`
`1366 GOTO 1367`
`1367 GOTO 1368`
`1368 GOTO 1369`
`1369 GOTO 1370`
`1370 GOTO 1371`
`1371 GOTO 1372`
`1372 GOTO 1373`
`1373 GOTO 1374`
`1374 GOTO 1375`
`1375 GOTO 1376`
`1376 GOTO 1377`
`1377 GOTO 1378`
`1378 GOTO 1379`
`1379 GOTO 1380`
`1380 GOTO 1381`
`1381 GOTO 1382`
`1382 GOTO 1383`
`1383 GOTO 1384`
`1384 GOTO 1385`
`1385 GOTO 1386`
`1386 GOTO 1387`
`1387 GOTO 1388`
`1388 GOTO 1389`
`1389 GOTO 1390`
`1390 GOTO 1391`
`1391 GOTO 1392`
`1392 GOTO 1393`
`1393 GOTO 1394`
`1394 GOTO 1395`
`1395 GOTO 1396`
`1396 GOTO 1397`
`1397 GOTO 1398`
`1398 GOTO 1399`
`1399 GOTO 1400`
`1400 GOTO 1401`
`1401 GOTO 1402`
`1402 GOTO 1403`
`1403 GOTO 1404`
`1404 GOTO 1405`
`1405 GOTO 1406`
`1406 GOTO 1407`
`1407 GOTO 1408`
`1408 GOTO 1409`
`1409 GOTO 1410`
`1410 GOTO 1411`
`1411 GOTO 1412`
`1412 GOTO 1413`
`1413 GOTO 1414`
`1414 GOTO 1415`
`1415 GOTO 1416`
`1416 GOTO 1417`
`1417 GOTO 1418`
`1418 GOTO 1419`
`1419 GOTO 1420`
`1420 GOTO 1421`
`1421 GOTO 1422`
`1422 GOTO 1423`
`1423 GOTO 1424`
`1424 GOTO 1425`
`1425 GOTO 1426`
`1426 GOTO 1427`
`1427 GOTO 1428`
`1428 GOTO 1429`
`1429 GOTO 1430`
`1430 GOTO 1431`
`1431 GOTO 1432`
`1432 GOTO 1433`
`1433 GOTO 1434`
`1434 GOTO 1435`
`1435 GOTO 1436`
`1436 GOTO 1437`
`1437 GOTO 1438`
`1438 GOTO 1439`
`1439 GOTO 1440`
`1440 GOTO 1441`
`1441 GOTO 1442`
`1442 GOTO 1443`
`1443 GOTO 1444`
`1444 GOTO 1445`
`1445 GOTO 1446`
`1446 GOTO 1447`
`1447 GOTO 1448`
`1448 GOTO 1449`
`1449 GOTO 1450`
`1450 GOTO 1451`
`1451 GOTO 1452`
`1452 GOTO 1453`
`1453 GOTO 1454`
`1454 GOTO 1455`
`1455 GOTO 1456`
`1456 GOTO 1457`
`1457 GOTO 1458`
`1458 GOTO 1459`
`1459 GOTO 1460`
`1460 GOTO 1461`
`1461 GOTO 1462`
`1462 GOTO 1463`
`1463 GOTO 1464`
`1464 GOTO 1465`
`1465 GOTO 1466`
`1466 GOTO 1467`
`1467 GOTO 1468`
`1468 GOTO 1469`
`1469 GOTO 1470`
`1470 GOTO 1471`
`1471 GOTO 1472`
`1472 GOTO 1473`
`1473 GOTO 1474`
`1474 GOTO 1475`
`1475 GOTO 1476`
`1476 GOTO 1477`
`1477 GOTO 1478`
`1478 GOTO 1479`
`1479 GOTO 1480`
`1480 GOTO 1481`
`1481 GOTO 1482`
`1482 GOTO 1483`
`1483 GOTO 1484`
`1484 GOTO 1485`
`1485 GOTO 1486`
`1486 GOTO 1487`
`1487 GOTO 1488`
`1488 GOTO 1489`
`1489 GOTO 1490`
`1490 GOTO 1491`
`1491 GOTO 1492`
`1492 GOTO 1493`
`1493 GOTO 1494`
`1494 GOTO 1495`
`1495 GOTO 1496`
`1496 GOTO 1497`
`1497 GOTO 1498`
`1498 GOTO 1499`
`1499 GOTO 1500`
`1500 GOTO 1501`
`1501 GOTO 1502`
`1502 GOTO 1503`
`1503 GOTO 1504`
`1504 GOTO 1505`
`1505 GOTO 1506`
`1506 GOTO 1507`
`1507 GOTO 1508`
`1508 GOTO 1509`
`1509 GOTO 1510`
`1510 GOTO 1511`
`1511 GOTO 1512`
`1512 GOTO 1513`
`1513 GOTO 1514`
`1514 GOTO 1515`
`1515 GOTO 1516`
`1516 GOTO 1517`
`1517 GOTO 1518`
`1518 GOTO 1519`
`1519 GOTO 1520`
`1520 GOTO 1521`
`1521 GOTO 1522`
`1522 GOTO 1523`
`1523 GOTO 1524`
`1524 GOTO 1525`
`1525 GOTO 1526`
`1526 GOTO 1527`
`1527 GOTO 1528`
`1528 GOTO 1529`
`1529 GOTO 1530`
`1530 GOTO 1531`
`1531 GOTO 1532`
`1532 GOTO 1533`
`1533 GOTO 1534`
`1534 GOTO 1535`
`1535 GOTO 1536`
`1536 GOTO 1537`
`1537 GOTO 1538`
`1538 GOTO 1539`
`1`

MEAN machine CODE

Programming in Basic is more restrictive than living on a student grant, but the prospect of learning machine language - with all those meaningless numbers and weird symbols - can seem daunting. But never fear, Jason Finch is here with a beginners' guide that's about as daunting as the Pepsi challenge.

PART ONE

The two words 'machine' and 'language', when used consecutively, can make even the sanest of people quiver at the knees, and can send less faints specimens crying for their mothers. I can even hear some of you screaming and handily trying to turn the page already, but don't, it will be worth the pain, the frustration, and the mental torment in the end - mastering machine code opens up limitless possibilities as far as programming goes.

For months you have seen Basic loaders in *Topical Type*, short programs full of BASIC statements which load some machine code into memory and

then do something rather

stunning. By the time this series has finished you will know exactly why they all work and you will know how to change them to do what you want. And now, using either the assembler on this month's package or the Action Replay cartridge, you'll get the chance throughout this series to mess about with real machine language, or ML, as it's known in the trade. For the moment I'll be referring to the AR card when I'm dealing with how to run the machine language listings, as it's slightly more straightforward to see what's going on, but from next month, I'll be joining the series loaders using 6502s. But feel free to experiment on the meantime.

THE APPROACH

It would help if you weren't just told there like a lesson reading this with no idea why. So I'll tell you the aim of all this and how I'm going to approach it. The aim is to get you programming in ML from day one, that's simple. I'll do it by telling you about the commands and the different ways of using the commands. I'll tell you about the concepts involved and about different memory locations. I won't assume you know anything about the type of programming at all, I

won't assume you're ever programmed a computer at all before, although I will make reference to Basic. I won't bore you with pointless details about hardware, about the CPU and about

WHAT IS... MACHINE LANGUAGE?

This seems like the most obvious question for a series like this. Machine language is a set of instructions that your computer understands without having to convert them into anything else. It is also a whole other way of thinking, so for us programming is concerned. Oh yes, forget about variables. The days of arrays and using many known letters of the alphabet have gone. (Things don't exist any more either. Say goodbye to your colors and your line numbers. The real programming is about to begin.

Other complicated issues that are of little relevance to the get-your-hands-to-work-on-it approach that I'm going to take. The deal is that I lecture and give examples, and you listen and test out the examples I give. Fair enough, yes?

YOUR FIRST TIME

The first time you do anything is always the worst time because you have to learn a load of things before you can even begin to do anything constructive. But hang on in there, it might seem a bit of an uphill struggle now, but persistence - the mastery of getting your name on the screen in Machine Language comes a nice second time in writing a really good program (er, yes, okay, I'll have to take your word on that one. - Dave).

In later months you'll be able to refer back to the

'What is...?' sections and really test what

WHAT IS... HEXADECIMAL?

It's even more complicated than binary, basically. Hexadecimal is a number system that uses the numbers 0 to 9 and the letters A to F. (BTW this includes letters as involved as well). The following table gives a comparison of decimal (decimal), and hex numbers. We'll use a dollar (\$) sign to show a hex number (it's the standard way).

Dec Hex	Dec Hex	Dec Hex	Dec Hex
0 \$0	4 \$4	8 \$8	12 \$C
1 \$1	5 \$5	9 \$9	13 \$D
2 \$2	6 \$6	10 \$A	14 \$E
3 \$3	7 \$7	11 \$B	15 \$F

You see, although hexadecimal has got more symbols than there are systems in most words (well the kind of words you get in the Sun, anyway), it's not all that complicated. Decimal 10 is hex \$A, and you fairly go missing from Basic. You can use hexadecimal without actually understanding how numbers are converted between it and decimal, assuming you have (give a calculation with that function, a cartridge with it or a silly little program like the ARB with one that I listed in *Topical Type*) is short-while-back. Hex is used mainly because it is convenient, single byte numbers are represented by two hex digits and double byte numbers are represented by four.

WHAT IS... A MEMORY LOCATION?

Although I have covered memory locations in *Dr Finch's Classroom*, I'm going to repeat them because the concept of memory locations is fundamental in learning machine language.

Numbers are stored in memory locations; these numbers are called bytes. The different memory locations have different functions and depending on the numbers stored in them, do different things. Ensure you understand memory locations before you continue with this series. Memory locations are also useful in ML, to store values because you do not have variables.



WHAT IS... A MACHINE LANGUAGE PROGRAM?

A machine language program is a list of instructions that is stored in memory from a particular memory location onwards. For example, you might start a Basic program at line 10, with a ML program you may start it at address 40450. This is a convenient memory location on the C64 and is \$C000 in hex.

you're looking for. The main fun will begin towards the end of this first part and by which time you'll have written your very first machine language program. It would help if you had an Action Replay cartridge — and I'm assuming you have one — purely for its machine language monitor, though. Most utility cartridges have monitors, so be prepared to use flexible if you don't actually own an A1 cartridge. It also has many references to things out of Dr Frost's Caseworkbook which I hope you have been following if not, well there are book reviews available, see page 26 — Ed.

THE FUN BEGINS

Okay, gently, hang on to your seats because the fun is about to begin. You've read all the 'What is...' bits, you've read all the relevant Dr Frost's Casework stuff, now you're ready to begin.

In ML, a load is an LD and a store is an ST. You follow this with the register

name that you want to load from or store to. So LDA, LDX, LDY and STA, STX, STY are all valid ML instructions. Loads need something to load and stores need somewhere to store things. Imagine you want to load the number 5 into the accumulator (the A register). In Basic you'd do `A=5` (because 5 is hex \$05, but because we want the NUMBER 5 and not MEMORY LOCATION 5, we put a hash sign before it: `#005`). This is the operand (see the 'What is...' bit on operands and operands with the brilliant accompanying picture).

The instruction to load something into the accumulator is LDA (Load) into the A register. So take `#005` as the first instruction. Now, we're actually got a machine language instruction/hex! What about storing? Well STA seems reasonable (Store the A register) but you need to specify somewhere to store it.

You give the instruction a memory address as the operand.

WHAT IS... AN OPERATOR AND AN OPERAND?

An operator is a thing that operates on certain bits of some particular job. An operand is the thing that is operated on.

When you get a jar of coffee at a stall, you are the operator. The coffee is the operand. Filtering is the thing in computer speak, the act of getting the coffee is the 'loading' and is split into knowing what you want — the coffee (the operand) — and you thinking about making for it (the operator).

\$C3,000 is \$0C00 in hex, and controls the border colour. So STX (\$0C00) could work, right?

The first thing we need to do is to stop the program if it does hops. In Basic this is done automatically, or with an END statement, in ML you use an instruction called RTS (Return from Subroutine) to get you out of a program.

HANDS-ON TIME

You want to write a machine language program consisting of the following three commands (yes you do, you know you do, go on admit it).

```
LDA #005
STX $0C00
RTS
```

In order to do this, you must assemble the program, and instruction at a time. The computer needs to know where you want to put this program in memory, and the most common place for beginners is \$C3,000 or \$C0000 in hex. The Action Replay monitor should be told the start address for

Best start! yay ah! Best start! yay ah!

WHAT IS... A REGISTER?

Good question, that one. A register is a place where things are stored, or registered. In machine language you have three registers, called the accumulator, the X-index and the Y-index. A, X and Y for short. Each register can store a one byte number. That is a number from 0 to 255.

assembling an instruction. It will then prompt you for the next instruction until you press Return without giving an instruction.

ASSEMBLING

Get your monitor up and running and enter A \$C000 LDA #005 and the Return. If you've done it right, you should see this on your screen:

```
>> 0000 A0 00 LDA #005
> 0000
```

Now, where the cursor is flashing, enter STA \$C000 and press the Return key, you enter RTS and press the Return key twice. You should have this on your screen now:

```
>> 0000 A0 00 LDA #005
>> 0000 80 00 STA $0000
>> 0000 00 00 RTS
>> 0000
```

You have now assembled a three-line machine language program. It has one load instruction, one store instruction, and one return. The numbers down the left are memory locations and the numbers immediately to their right are the values stored at those locations. So, 0A0 (\$0A in decimal) is stored at \$C000 (40960 in decimal), 80 is stored at the next location, 00 at the next, then 00, 00 and finally 00 is stored at \$C003.

These all mean something (and I thought they were just there to confuse me — Dave). 0A0 is the code for our particular LDA instruction, 80 is the operand, of course, 00 is the code for our STA instruction, and 00.



WHAT IS... BINARY?

If you haven't been following Dr Frost's Casework then you have committed the greatest of sins, you haven't learned about binary. Binary is a number system that uses only 0s and 1s. The following table gives a comparison of decimal (base 10) and binary numbers. We'll use a percent (%) sign to show a binary number.

Dec	Bin	Dec	Bin
0	%0000	1	%0001
2	%0010	3	%0011
4	%0100	5	%0101
6	%0110	7	%0111
8	%1000	9	%1001
10	%1010	11	%1011
12	%1100	13	%1101
14	%1110	15	%1111

(b) is a two-byte address. You should remember this notation from the "What is a type?" question. The `0010` is the code for `int`. You do not necessarily remember those codes; they are the raw machine code – the number on the values that are `P(0)(0)` into memory, which also are a `short` (short is native code).

RUNNING

To run that machine-language program, you enter the G prompt, for "Go." Register G 000000 has engineering flag, so the cursor has turned left. You're not particularly excited, I can tell. Perhaps you can turn the background light so as well to make you a bit more happy. With the cursor flashing after a job, enter 4, 0004, 000, 0000 and press the Return key. Now enter 0000, 000001, press the Return key, enter RTN and press the Return key. You should get:

1. 1997. 40 000 000 000 000
 2. 1997 00 01 00 00 00 00 00 00
 3. 1997 40 00 00 00
 4. 1997

**WHAT IS...
A MACHINE**

WHAT IS... A MACHINE LANGUAGE MONITOR?

A machine language monitor, or MML, is a program that allows you to write and view machine language programs. They often do other things as well, but they are secondary. The Action Replay cartridge has a MML built into it which can be accessed from Basic by either pressing F8, entering MML and pressing the Enter/Return or select MML from the option screen that comes up when you press the controller button on the back. You should see something like this come up on your screen:

Year	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099
1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	

This number will be flashing red to the cops - you have entered the realm of ML. No-one can hear you scream! I suggest you read the little number of your manual for work and quarterly time to use it. For life events, order a new manual. Please do not have to read.

put it into the accumulator. You then take the number in the accumulator and store it at memory location 00,000. That changes the **border color** to the same as any 00000 00,000, it would 000 in Basic. The next action, with the second kind and store, you take the number 8 again and put it into the accumulator. You then store that at location 00,001, changing the **background color**. Check out the **color codes** in the table below and choose the **hex** to development.

[illegible]

804 purple	802 medium grey
806 green	803 light green
808 blue	805 light blue
807 yellow	801 light grey

DIVERSITY MATTERS

As a person disconnected, every action has its opposite and equal reaction, and the opposite of assembling is disassembling. You probably won't be too surprised to find, after reading that you've been slightly following all the instructions I've been giving you on the mail I like to think there is some purpose in my life within your AP number. (note I C000) C000 and you should see:

```

..: 0000 0000
..: 0000 00 00      L20 0000
..: 0000 00 00 00   000 0000

```

This should not come as a great shock to you because that is what you originally entered (I, as I said, you've been following my instructions, otherwise you'd know what you might get - some rude error message about Intel). This is part of your machine's hardware program.

You can go up and edit any of the lines. It works the same way you would with a basic program. For example, move the cursor over the second C of the

WHAT IS... A RITE?

apart from being the cause of many a hilarious joke in early file systems (in comparison with the many suitably named computers there were about then), what is a byte? Your computer has 65,536 different memory locations. Think of them as boxes in which you can place bytes of data. 65,536 in a memory location that just happens to control the border colour. PCMCIA knows it's safe to block. One byte can be shared at most location. A byte, on the CPU, is a number between 0 and 255. Millions. 255 in decimal is 11111111 in binary. 0 is 00000000.

A two-byte number is said which takes up two bytes in memory (effectively), consisting from 256 up to 65,535 (two bytes = a "word" byte as a single byte). To get the first number, you take the high byte, multiply it by 256 and add the low byte. So, for example, 55,238 is stored with high byte (55) and low byte (238) because $256 \times 55 + 238 = 55,238$. In your C# they are stored the other way around, to be consistent, so the number 55,238 would be stored as 238,55 in memory. In fact, these are 65535 and 65536 in decimal. 65535 is $2^{16}-1$.

Check out the similarity with the high byte and the low byte conversions. That's why he is so concerned. It's a lot easier to see that the high byte of 0000 is 000 than it is to work in decimal and say the high byte of 00 0000 is 000.

[illegible][illegible]

WHAT IS... LOADING AND STORING?

Imagine you are in a supermarket and one of the things you want is a box of cereal. You pick up the cereal and you put it in your trolley. (Change, yes, I can deal, an extended metaphor opening on cereal – I love it. This is loading and storing. Loading is the action of getting the cereal, and storing is the action of putting it in the trolley. The cereal happens in *Store*, where you do *PUT* 51189). In The computer 'loads' something or other with the number 51 and then 'stores' it at location 51189. It is simply another for the language of operations.

