

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**  
**по практической работе №1**  
**по дисциплине «Операционные системы»**  
**Тема: Исследование структур загрузочных модулей**

Студентка гр. 8381

\_\_\_\_\_

Звегинцева Е.Н.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

### **Цель работы.**

Исследование различий в структурах исходных текстов модулей .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

### **Необходимые сведения для составления программы.**

Тип IBM PC хранится в байте по адресу 0F000:0FFFE, в предпоследнем байте ROM BIOS. Соответствие кода и типа в таблице:

<b>PC</b>	<b>FF</b>
<b>PC/XT</b>	<b>FE,FB</b>
<b>AT</b>	<b>FC</b>
<b>PS2 модель 30</b>	<b>FA</b>
<b>PS2 модель 50 или 60</b>	<b>FC</b>
<b>PS2 модель 80</b>	<b>F8</b>
<b>PCjr</b>	<b>FD</b>
<b>PC Convertible</b>	<b>F9</b>

Для определения версии MS DOS следует воспользоваться функцией 30H прерывания 21H. Входным параметром является номер функции в AH:

Выходными параметрами являются:

AL – номер основной версии. Если 0, то <2.0;

AH – номер модификации;

BH – серийный номер OEM (Original Equipment Manufacturer);

BL:CH – 24-битовый серийный номер пользователя.

### **Постановка задачи.**

Требуется реализовать текст исходного .COM модуля, который определяет тип PC и версию системы. Ассемблерная программа должна читать содержимое предпоследнего байта ROM BIOS, по таблице, сравнивая коды, определять тип PC и выводить строку с названием модели. Если код не совпадает ни с одним значением, то двоичный код переводиться в символьную строку, содержащую запись шестнадцатеричного числа и выводиться на экран в виде соответствующего сообщения. Затем определяется версия системы. Ассемблерная программа должна по значениям регистров AL и AH формировать текстовую строку в формате xx.yy, где xx - номер основной версии, а yy - номер модификации в десятичной системе счисления, формировать строки с серийным номером OEM (Original Equipment Manufacturer) и серийным номером пользователя. Полученные строки выводятся на экран.

Далее необходимо отладить полученный исходный модуль и получить «хороший» .COM модуль, а также необходимо построить «плохой» .EXE, полученный из исходного текста для .COM модуля.

Затем нужно написать текст «хорошего» .EXE модуля, который выполняет те же функции, что и модуль .COM, далее его построить, отладить и сравнить исходные тексты для .COM и .EXE модулей.

### **Выполнение работы.**

Выполнение работы производилось на базе операционной системы Windows 10. Сборка и отладка модулей производилась с помощью компиляторов MASM и TASM и отладчика TD в эмуляторе DOSBox.

Типы исполняемых файлов в DOS исследовались на примере программы, которая выводит версию операционной системы. Программа содержит таблицу кодов, обозначающих различные типы PC. Она считывает код из памяти BIOS, ищет его по таблице и выводит строку с названием этого типа (функции LOOKUP\_MODEL и PRINT\_MODEL). Затем загружается информация о версии DOS, и отображается в функции

PRINT\_VERSION\_INFO. Вывод чисел выполняется при помощи функций печати чисел в строку BYTE\_TO\_DEC, BYTE\_TO\_HEX и WRD\_TO\_HEX из методички. Эта программа была написана таким образом, чтобы из неё можно было получить загрузочный модуль типа COM (LAB1COM.ASM). После этого она была модифицирована в программу EXE (LAB1EXE.ASM).

Пример сборки .COM модуля показан на рис. 1.

```
C:\>tasm lab1com.asm
Turbo Assembler Version 4.0 Copyright (c) 1988, 1993 Borland International

Assembling file:   lab1com.asm
Error messages:    None
Warning messages:  None
Passes:           1
Remaining memory:  466k

C:\>tlink /t lab1com.obj
Turbo Link Version 4.01 Copyright (c) 1991 Borland International
```

Рисунок 1 – Полученный .COM модуль

Во время линковки «плохого» .EXE модуля было выведено предупреждение об отсутствии сегмента стека, представлено на рис. 2.

Запуск «плохого» .EXE модуля.

```
C:\>link lab1com.obj

Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

Run File [LAB1COM.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
```

Рисунок 2 – Предупреждение во время линковки «плохого» .EXE

```

C:\>lab1com.exe
= f 0 i 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99 101 103 105 107 109 111 113 115 117 119 121 123 125 127 129 131 133 135 137 139 141 143 145 147 149 151 153 155 157 159 161 163 165 167 169 171 173 175 177 179 181 183 185 187 189 191 193 195 197 199 201 203 205 207 209 211 213 215 217 219 221 223 225 227 229 231 233 235 237 239 241 243 245 247 249 251 253 255 257 259 261 263 265 267 269 271 273 275 277 279 281 283 285 287 289 291 293 295 297 299 301 303 305 307 309 311 313 315 317 319 321 323 325 327 329 331 333 335 337 339 341 343 345 347 349 351 353 355 357 359 361 363 365 367 369 371 373 375 377 379 381 383 385 387 389 391 393 395 397 399 401 403 405 407 409 411 413 415 417 419 421 423 425 427 429 431 433 435 437 439 441 443 445 447 449 451 453 455 457 459 461 463 465 467 469 471 473 475 477 479 481 483 485 487 489 491 493 495 497 499 501 503 505 507 509 511 513 515 517 519 521 523 525 527 529 531 533 535 537 539 541 543 545 547 549 551 553 555 557 559 561 563 565 567 569 571 573 575 577 579 581 583 585 587 589 591 593 595 597 599 601 603 605 607 609 611 613 615 617 619 621 623 625 627 629 631 633 635 637 639 641 643 645 647 649 651 653 655 657 659 661 663 665 667 669 671 673 675 677 679 681 683 685 687 689 691 693 695 697 699 701 703 705 707 709 711 713 715 717 719 721 723 725 727 729 731 733 735 737 739 741 743 745 747 749 751 753 755 757 759 761 763 765 767 769 771 773 775 777 779 781 783 785 787 789 791 793 795 797 799 801 803 805 807 809 811 813 815 817 819 821 823 825 827 829 831 833 835 837 839 841 843 845 847 849 851 853 855 857 859 861 863 865 867 869 871 873 875 877 879 881 883 885 887 889 891 893 895 897 899 901 903 905 907 909 911 913 915 917 919 921 923 925 927 929 931 933 935 937 939 941 943 945 947 949 951 953 955 957 959 961 963 965 967 969 971 973 975 977 979 981 983 985 987 989 991 993 995 997 999 1001 1003 1005 1007 1009 1011 1013 1015 1017 1019 1021 1023 1025 1027 1029 1031 1033 1035 1037 1039 1041 1043 1045 1047 1049 1051 1053 1055 1057 1059 1061 1063 1065 1067 1069 1071 1073 1075 1077 1079 1081 1083 1085 1087 1089 1091 1093 1095 1097 1099 1101 1103 1105 1107 1109 1111 1113 1115 1117 1119 1121 1123 1125 1127 1129 1131 1133 1135 1137 1139 1141 1143 1145 1147 1149 1151 1153 1155 1157 1159 1161 1163 1165 1167 1169 1171 1173 1175 1177 1179 1181 1183 1185 1187 1189 1191 1193 1195 1197 1199 1201 1203 1205 1207 1209 1211 1213 1215 1217 1219 1221 1223 1225 1227 1229 1231 1233 1235 1237 1239 1241 1243 1245 1247 1249 1251 1253 1255 1257 1259 1261 1263 1265 1267 1269 1271 1273 1275 1277 1279 1281 1283 1285 1287 1289 1291 1293 1295 1297 1299 1301 1303 1305 1307 1309 1311 1313 1315 1317 1319 1321 1323 1325 1327 1329 1331 1333 1335 1337 1339 1341 1343 1345 1347 1349 1351 1353 1355 1357 1359 1361 1363 1365 1367 1369 1371 1373 1375 1377 1379 1381 1383 1385 1387 1389 1391 1393 1395 1397 1399 1401 1403 1405 1407 1409 1411 1413 1415 1417 1419 1421 1423 1425 1427 1429 1431 1433 1435 1437 1439 1441 1443 1445 1447 1449 1451 1453 1455 1457 1459 1461 1463 1465 1467 1469 1471 1473 1475 1477 1479 1481 1483 1485 1487 1489 1491 1493 1495 1497 1499 1501 1503 1505 1507 1509 1511 1513 1515 1517 1519 1521 1523 1525 1527 1529 1531 1533 1535 1537 1539 1541 1543 1545 1547 1549 1551 1553 1555 1557 1559 1561 1563 1565 1567 1569 1571 1573 1575 1577 1579 1581 1583 1585 1587 1589 1591 1593 1595 1597 1599 1601 1603 1605 1607 1609 1611 1613 1615 1617 1619 1621 1623 1625 1627 1629 1631 1633 1635 1637 1639 1641 1643 1645 1647 1649 1651 1653 1655 1657 1659 1661 1663 1665 1667 1669 1671 1673 1675 1677 1679 1681 1683 1685 1687 1689 1691 1693 1695 1697 1699 1701 1703 1705 1707 1709 1711 1713 1715 1717 1719 1721 1723 1725 1727 1729 1731 1733 1735 1737 1739 1741 1743 1745 1747 1749 1751 1753 1755 1757 1759 1761 1763 1765 1767 1769 1771 1773 1775 1777 1779 1781 1783 1785 1787 1789 1791 1793 1795 1797 1799 1801 1803 1805 1807 1809 1811 1813 1815 1817 1819 1821 1823 1825 1827 1829 1831 1833 1835 1837 1839 1841 1843 1845 1847 1849 1851 1853 1855 1857 1859 1861 1863 1865 1867 1869 1871 1873 1875 1877 1879 1881 1883 1885 1887 1889 1891 1893 1895 1897 1899 1901 1903 1905 1907 1909 1911 1913 1915 1917 1919 1921 1923 1925 1927 1929 1931 1933 1935 1937 1939 1941 1943 1945 1947 1949 1951 1953 1955 1957 1959 1961 1963 1965 1967 1969 1971 1973 1975 1977 1979 1981 1983 1985 1987 1989 1991 1993 1995 1997 1999 2001 2003 2005 2007 2009 2011 2013 2015 2017 2019 2021 2023 2025 2027 2029 2031 2033 2035 2037 2039 2041 2043 2045 2047 2049 2051 2053 2055 2057 2059 2061 2063 2065 2067 2069 2071 2073 2075 2077 2079 2081 2083 2085 2087 2089 2091 2093 2095 2097 2099 2101 2103 2105 2107 2109 2111 2113 2115 2117 2119 2121 2123 2125 2127 2129 2131 2133 2135 2137 2139 2141 2143 2145 2147 2149 2151 2153 2155 2157 2159 2161 2163 2165 2167 2169 2171 2173 2175 2177 2179 2181 2183 2185 2187 2189 2191 2193 2195 2197 2199 2201 2203 2205 2207 2209 2211 2213 2215 2217 2219 2221 2223 2225 2227 2229 2231 2233 2235 2237 2239 2241 2243 2245 2247 2249 2251 2253 2255 2257 2259 2261 2263 2265 2267 2269 2271 2273 2275 2277 2279 2281 2283 2285 2287 2289 2291 2293 2295 2297 2299 2301 2303 2305 2307 2309 2311 2313 2315 2317 2319 2321 2323 2325 2327 2329 2331 2333 2335 2337 2339 2341 2343 2345 2347 2349 2351 2353 2355 2357 2359 2361 2363 2365 2367 2369 2371 2373 2375 2377 2379 2381 2383 2385 2387 2389 2391 2393 2395 2397 2399 2401 2403 2405 2407 2409 2411 2413 2415 2417 2419 2421 2423 2425 2427 2429 2431 2433 2435 2437 2439 2441 2443 2445 2447 2449 2451 2453 2455 2457 2459 2461 2463 2465 2467 2469 2471 2473 2475 2477 2479 2481 2483 2485 2487 2489 2491 2493 2495 2497 2499 2501 2503 2505 2507 2509 2511 2513 2515 2517 2519 2521 2523 2525 2527 2529 2531 2533 2535 2537 2539 2541 2543 2545 2547 2549 2551 2553 2555 2557 2559 2561 2563 2565 2567 2569 2571 2573 2575 2577 2579 2581 2583 2585 2587 2589 2591 2593 2595 2597 2599 2601 2603 2605 2607 2609 2611 2613 2615 2617 2619 2621 2623 2625 2627 2629 2631 2633 2635 2637 2639 2641 2643 2645 2647 2649 2651 2653 2655 2657 2659 2661 2663 2665 2667 2669 2671 2673 2675 2677 2679 2681 2683 2685 2687 2689 2691 2693 2695 2697 2699 2701 2703 2705 2707 2709 2711 2713 2715 2717 2719 2721 2723 2725 2727 2729 2731 2733 2735 2737 2739 2741 2743 2745 2747 2749 2751 2753 2755 2757 2759 2761 2763 2765 2767 2769 2771 2773 2775 2777 2779 2781 2783 2785 2787 2789 2791 2793 2795 2797 2799 2801 2803 2805 2807 2809 2811 2813 2815 2817 2819 2821 2823 2825 2827 2829 2831 2833 2835 2837 2839 2841 2843 2845 2847 2849 2851 2853 2855 2857 2859 2861 2863 2865 2867 2869 2871 2873 2875 2877 2879 2881 2883 2885 2887 2889 2891 2893 2895 2897 2899 2901 2903 2905 2907 2909 2911 2913 2915 2917 2919 2921 2923 2925 2927 2929 2931 2933 2935 2937 2939 2941 2943 2945 2947 2949 2951 2953 2955 2957 2959 2961 2963 2965 2967 2969 2971 2973 2975 2977 2979 2981 2983 2985 2987 2989 2991 2993 2995 2997 2999 3001 3003 3005 3007 3009 3011 3013 3015 3017 3019 3021 3023 3025 3027 3029 3031 3033 3035 3037 3039 3041 3043 3045 3047 3049 3051 3053 3055 3057 3059 3061 3063 3065 3067 3069 3071 3073 3075 3077 3079 3081 3083 3085 3087 3089 3091 3093 3095 3097 3099 3101 3103 3105 3107 3109 3111 3113 3115 3117 3119 3121 3123 3125 3127 3129 3131 3133 3135 3137 3139 3141 3143 3145 3147 3149 3151 3153 3155 3157 3159 3161 3163 3165 3167 3169 3171 3173 3175 3177 3179 3181 3183 3185 3187 3189 3191 3193 3195 3197 3199 3201 3203 3205 3207 3209 3211 3213 3215 3217 3219 3221 3223 3225 3227 3229 3231 3233 3235 3237 3239 3241 3243 3245 3247 3249 3251 3253 3255 3257 3259 3261 3263 3265 3267 3269 3271 3273 3275 3277 3279 3281 3283 3285 3287 3289 3291 3293 3295 3297 3299 3301 3303 3305 3307 3309 3311 3313 3315 3317 3319 3321 3323 3325 3327 3329 3331 3333 3335 3337 3339 3341 3343 3345 3347 3349 3351 3353 3355 3357 3359 3361 3363 3365 3367 3369 3371 3373 3375 3377 3379 3381 3383 3385 3387 3389 3391 3393 3395 3397 3399 3401 3403 3405 3407 3409 3411 3413 3415 3417 3419 3421 3423 3425 3427 3429 3431 3433 3435 3437 3439 3441 3443 3445 3447 3449 3451 3453 3455 3457 3459 3461 3463 3465 3467 3469 3471 3473 3475 3477 3479 3481 3483 3485 3487 3489 3491 3493 3495 3497 3499 3501 3503 3505 3507 3509 3511 3513 3515 3517 3519 3521 3523 3525 3527 3529 3531 3533 3535 3537 3539 3541 3543 3545 3547 3549 3551 3553 3555 3557 3559 3561 3563 3565 3567 3569 3571 3573 3575 3577 3579 3581 3583 3585 3587 3589 3591 3593 3595 3597 3599 3601 3603 3605 3607 3609 3611 3613 3615 3617 3619 3621 3623 3625 3627 3629 3631 3633 3635 3637 3639 3641 3643 3645 3647 3649 3651 3653 3655 3657 3659 3661 3663 3665 3667 3669 3671 3673 3675 3677 3679 3681 3683 3685 3687 3689 3691 3693 3695 3697 3699 3701 3703 3705 3707 3709 3711 3713 3715 3717 3719 3721 3723 3725 3727 3729 3731 3733 3735 3737 3739 3741 3743 3745 3747 3749 3751 3753 3755 3757 3759 3761 3763 3765 3767 3769 3771 3773 3775 3777 3779 3781 3783 3785 3787 3789 3791 3793 3795 3797 3799 3801 3803 3805 3807 3809 3811 3813 3815 3817 3819 3821 3823 3825 3827 3829 3831 3833 3835 3837 3839 3841 3843 3845 3847 3849 3851 3853 3855 3857 3859 3861 3863 3865 3867 3869 3871 3873 3875 3877 3879 3881 3883 3885 3887 3889 3891 3893 3895 3897 3899 3901 3903 3905 3907 3909 3911 3913 3915 3917 3919 3921 3923 3925 3927 3929 3931 3933 3935 3937 3939 3941 3943 3945 3947 3949 3951 3953 3955 3957 3959 3961 3963 3965 3967 3969 3971 3973 3975 3977 3979 3981 3983 3985 3987 3989 3991 3993 3995 3997 3999 4001 4003 4005 4007 4009 4011 4013 4015 4017 4019 4021 4023 4025 4027 4029 4031 4033 4035 4037 4039 4041 4043 4045 4047 4049 4051 4053 4055 4057 4059 4061 4063 4065 4067 4069 4071 4073 4075 4077 4079 4081 4083 4085 4087 4089 4091 4093 4095 4097 4099 4101 4103 4105 4107 4109 4111 4113 4115 4117 4119 4121 4123 4125 4127 4129 4131 4133 4135 4137 4139 4141 4143 4145 4147 4149 4151 4153 4155 4157 4159 4161 4163 4165 4167 4169 4171 4173 4175 4177 4179 4181 4183 4185 4187 4189 4191 4193 4195 4197 4199 4201 4203 4205 4207 4209 4211 4213 4215 4217 4219 4221 4223 4225 4227 4229 4231 4233 4235 4237 4239 4241 4243 4245 4247 4249 4251 4253 4255 4257 4259 4261 4263 4265 4267 4269 4271 4273 4275 4277 4279 4281 4283 4285 4287 4289 4291 4293 4295 4297 4299 4301 4303 4305 4307 4309 4311 4313 4315 4317 4319 4321 4323 4325 4327 4329 4331 4333 4335 4337 4339 4341 4343 4345 4347 4349 4351 4353 4355 4357 4359 4361 4363 4365 4367 4369 4371 4373 4375 4377 4379 4381 4383 4385 4387 4389 4391 4393 4395 4397 4399 4401 4403 4405 4407 4409 4411 4413 4415 4417 4419 4421 4423 4425 4427 4429 4431 4433 4435 4437 4439 4441 4443 4445 4447 4449 4451 4453 4455 4457 4459 4461 4463 4465 4467 4469 4471 4473 4475 4477 4479 4481 4483 4485 4487 4489 4491 4493 4495 4497 4499 4501 4503 4505 4507 4509 4511 4513 4515 4517 4519 4521 4523 4525 4527 4529 4531 4533 4535 4537 4539 4541 4543 4545 4547 4549 4551 4553 4555 4557 4559 4561 4563 4565 4567 4569 4571 4573 4575 4577 4579 4581 4583 4585 4587 4589 4591 4593 4595 4597 4599 4601 4603 4605 4607 4609 4611 4613 4615 4617 4619 4621 4623 4625 4627 4629 4631 4633 4635 4637 4639 4641 4643 4645 4647 4649 4651 4653 4655 4657 4659 4661 4663 4665 4667 4669 4671 4673 4675 4677 4679 4681 4683 4685 4687 4689 4691 4693 4695 4697 4699 4701 4703 4705 4707 4709 4711 4713 4715 4717 4719 4721 4723 4725 4727 4729 4731 4733 4735 4737 4739 4741 4743 4745 4747 4749 4751 4753 4755 4757 4759 4761 4763 4765 4767 4769 4771 4773 4775 4777 4779 4781 4783 4785 4787 4789 4791 4793 4795 4797 4799 4801 4803 4805 4807 4809 4811 4813 4815 4817 4819 4821 4823 4825 4827 4829 4831 4833 4835 4837 4839 4841 4843 4845 4847 4849 4851 4853 4855 4857 4859 4861 4863 4865 4867 4869 4871 4873 4875 4877 4879 4881 4883 4885 4887 4889 4891 4893 4895 4897 4899 4901 4903 4905 4907 4909 4911 4913 4915 4917 4919 4921 4923 4925 4927 4929 4931 4933 4935 4937 4939 4941 4943 4945 4947 4949 4951 4953 4955 4957 4959 4961 4963 4965 4967 4969 4971 4973 4975 4977 4979 4981 4983 4985 4987 4989 4991 4993 4995 4997 4999 5001 5003 5005 5007 5009 5011 5013 5015 5017 5019 5021 5023 5025 5027 5029 5031 5033 5035 5037 5039 5041 5043 5045 5047 5049 5051 5053 5055 5057 5059 5061 5063 5065 5067 5069 5071 5073 5075 5077 5079 5081 5083 5085 5087 5089 5091 5093 5095 5097 5099 5101 5103 5105 5107 5109 5111 5113 5115 5117 5119 5121 5123 5125 5127 5129 5131 5133 5135 5137 5139 5141 5143 5145 5147 5149 5151 5153 5155 51
```

располагает код программы только после этого блока. Также обязательна директива ASSUME, которая устанавливает значения регистров CS, DS на адрес сегмента программы.

#### 4. Все ли форматы команд можно использовать в COM-программе?

Нельзя использовать команды, связанные с адресом сегмента, так как он неизвестен до загрузки этого сегмента в память, т.к. в .COM файле отсутствует таблица настройки с информацией о типе адресов и их местоположении.

### Отличия форматов файлов COM и EXE модулей.

#### 1. Какова структура файла COM? С какого адреса располагается код?

Вид файла COM в шестнадцатеричном формате представлен на рис. 6.

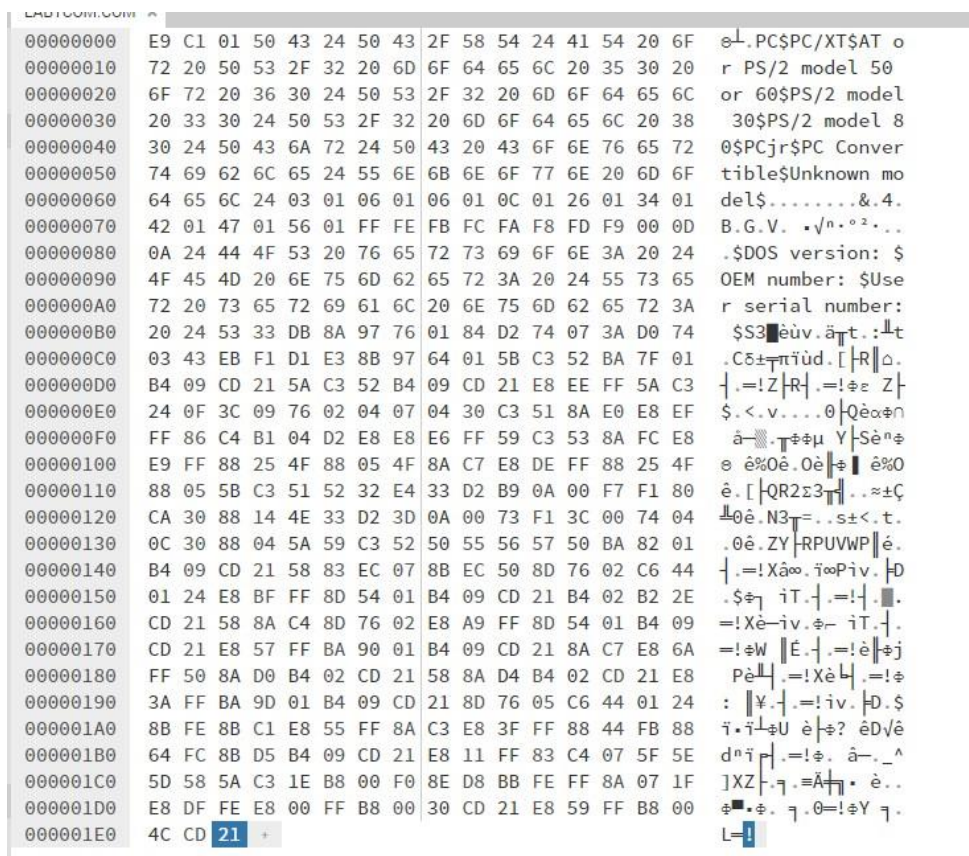


Рисунок 6 – Вид COM файла в шестнадцатеричном виде

COM файл состоит из одного сегмента и содержит данные и машинные команды. размер файла не превышает 64 КБ.

Файл начинается с E9 C1 01 – jmp к адресу 0x103 + 0x1c1 (0x103 – значение IP при выполнении этой команды), с которого начинается функция

MAIN. После кода программы в файле также ничего нет. По файлу карты памяти, который выдаёт линковщик, видно, что длина файла 1E316 равна длине единственного сегмента за вычетом длины PSP (10016 байт)

Start	Stop	Length	Name	Class
-------	------	--------	------	-------

000000h	002E2Nh	002E3H	TEXT	
---------	---------	--------	------	--

## 2. Какова структура файла «плохого» EXE? С какого адреса располагается код? Что располагается с 0 адреса?

Вид «плохого» EXE файла в шестнадцатеричном формате представлен на рис. 7.

00000000	4D 5A E3 00 03 00 00 00	20 00 00 00 FF FF 00 00	.....
00000010	00 00 50 E0 00 01 00 00	1E 00 00 00 01 00 00 00	..Pα.....
00000020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
00000030	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
00000040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
00000050	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
00000060	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
00000070	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
00000080	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
00000090	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000000A0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000000B0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000000C0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000000D0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000000E0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000000F0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
00000100	E9 C1 01 50 43 24 50 43	2F 58 54 24 41 54 20 6F	.....PC\$PC/XT\$AT o
00000110	72 20 50 53 2F 32 20 6D	6F 64 65 6C 20 35 30 20	r PS/2 model 50
00000120	6F 72 20 36 30 24 50 53	2F 32 20 6D 6F 64 65 6C	or 60\$PS/2 model
00000130	20 33 30 24 50 53 2F 32	20 6D 6F 64 65 6C 20 38	30\$PS/2 model 8
00000140	30 24 50 43 6A 72 24 50	43 20 43 6F 6E 76 65 72	0\$PCjr\$PC Conver
00000150	74 69 62 6C 65 24 55 6E	6B 6E 6F 77 6E 20 6D 6F	tible\$Unknown mo
00000160	64 65 6C 24 03 01 06 01	06 01 0C 01 26 01 34 01	del\$.&.4.
00000170	42 01 47 01 56 01 FF FE	FB FC FA F8 FD F9 00 0D	B.G.V. .√n.°2. .
00000180	0A 24 44 4F 53 20 76 65	72 73 69 6F 6E 3A 20 24	.\$DOS version: \$
00000190	4F 45 4D 20 6E 75 6D 62	65 72 3A 20 24 55 73 65	OEM number: \$Use
000001A0	72 20 73 65 72 69 61 6C	20 6E 75 6D 62 65 72 3A	r serial number:
000001B0	20 24 53 33 DB 8A 97 76	01 84 D2 74 07 3A D0 74	\$S3\$èùv.äTt. :Lt
000001C0	03 43 EB F1 D1 E3 8B 97	64 01 5B C3 52 BA 7F 01	.C5±πiüd.[ R  .o.
000001D0	B4 09 CD 21 5A C3 52 B4	09 CD 21 E8 EE FF 5A C3	-. =!Z R . =!φε Z
000001E0	24 0F 3C 09 76 02 04 07	04 30 C3 51 8A E0 E8 EF	\$.<.v...0 Qèαφn
000001F0	FF 86 C4 B1 04 D2 E8 E8	E6 FF 59 C3 53 8A FC E8	ä-  .Tφμ Y Sèñφ
00000200	E9 FF 88 25 4F 88 05 4F	8A C7 E8 DE FF 88 25 4F	ε è%0è.0è  φ    è%0
00000210	88 05 5B C3 51 52 32 E4	33 D2 B9 0A 00 F7 F1 80	è.[ QR2z3T  .≈±Ç
00000220	CA 30 88 14 4E 33 D2 3D	0A 00 73 F1 3C 00 74 04	Δ0è.N3T=..s<.t.
00000230	0C 30 88 04 5A 59 C3 52	50 55 56 57 50 BA 82 01	.0è.ZY RPUVWP  é.
00000240	B4 09 CD 21 58 83 EC 07	8B EC 50 8D 76 02 C6 44	-. =!Xä∞.i∞Piv.†D
00000250	01 24 E8 BF FF 8D 54 01	B4 09 CD 21 B4 02 B2 2E	.\$φγ iT. - =! .  .
00000260	CD 21 58 8A C4 8D 76 02	E8 A9 FF 8D 54 01 B4 09	=!Xè-iv.φ- iT. -.
00000270	CD 21 E8 57 FF BA 90 01	B4 09 CD 21 8A C7 E8 6A	=!φW   E. - =!è  φj
00000280	FF 50 8A D0 B4 02 CD 21	58 8A D4 B4 02 CD 21 E8	Pè  . =!Xè  . =!φ
00000290	3A FF BA 9D 01 B4 09 CD	21 8D 76 05 C6 44 01 24	:   Y. - =!iv.†D.\$
000002A0	8B FE 8B C1 E8 55 FF 8A	C3 E8 3F FF 88 44 FB 88	i.~iLφU è φ? èDvè
000002B0	64 FC 8B D5 B4 09 CD 21	E8 11 FF 83 C4 07 5F 5E	d^n i -. =!φ. ä-..^
000002C0	5D 58 5A C3 1E B8 00 F0	8E D8 BB FE FF 8A 07 1F	]XZ -. =!A  . è..
000002D0	E8 DF FE E8 00 FF B8 00	30 CD 21 E8 59 FF B8 00	φ  .φ. γ.0=!!φY γ.
000002E0	4C CD 21		L=!

Рисунок 7 – Вид «плохого» EXE файла

В «плохом» EXE файле данные и код содержатся в одном сегменте. Код располагается с адреса 300h. С адреса 0h располагается управляющая



информация для загрузчика(таблица настроек и др.информация).

В "плохом" ехе резервируются дополнительные 100h с помощью команды ORG 100h, которые в сом требуются для PSP, поэтому адреса начала кода в "хорошем" и "плохом" ехе отличаются на 100h+S, где S – размер стека.

### 3. Какова структура файла «хорошего» EXE? Чем он отличается от «плохого» EXE файла?

Вид «хорошего» EXE в шестнадцатеричном виде представлен на рис.8

-без_названия- x	LAB EXE.EXE x	
00000000	4D 5A E6 01 03 00 01 00	20 00 00 00 FF FF 00 00 MZu..... ..
00000010	00 02 CA 66 12 01 2B 00	1E 00 00 00 01 00 17 01 ..lf...+.....
00000020	2B 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 +.....
00000030	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 .....
00000040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 .....
00000050	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 .....
00000060	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 .....
00000070	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 .....
00000080	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 .....
00000090	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 .....
000003C0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 .....
000003D0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 .....
000003E0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 .....
000003F0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 .....
00000400	50 43 24 50 43 2F 58 54	24 41 54 20 6F 72 20 50 PC\$PC/XT\$AT or P
00000410	53 2F 32 20 6D 6F 64 65	6C 20 35 30 20 6F 72 20 S/2 model 50 or
00000420	36 30 24 50 53 2F 32 20	6D 6F 64 65 6C 20 33 30 60\$PS/2 model 30
00000430	24 50 53 2F 32 20 6D 6F	64 65 6C 20 38 30 24 50 \$PS/2 model 80\$P
00000440	43 6A 72 24 50 43 20 43	6F 6E 76 65 72 74 69 62 Cjr\$PC Convertib
00000450	6C 65 24 55 6E 68 6E 6F	77 6E 20 6D 6F 64 65 6C le\$Unknown model
00000460	24 00 00 03 00 03 00 09	00 23 00 31 00 3F 00 44 \$......#.1.?.D
00000470	00 53 00 FF FE FB FC FA	F8 FD F9 00 0D 0A 24 44 .S. .√n.°²...\$D
00000480	4F 53 20 76 65 72 73 69	6F 6E 3A 20 24 4F 45 4D OS version: \$OEM
00000490	20 6E 75 6D 62 65 72 3A	20 24 55 73 65 72 20 73 number: \$User s
000004A0	65 72 69 61 6C 20 6E 75	6D 62 65 72 3A 20 24 00 erial number: \$.
000004B0	53 33 DB 8A 97 73 00 84	D2 74 07 3A D0 74 03 43 S3ëùs.äTt.:!t.C
000004C0	EB F1 D1 E3 8B 97 61 00	5B C3 52 BA 7C 00 B4 09 5±πiua.[ R  . .
000004D0	CD 21 5A C3 52 B4 09 CD	21 E8 EE FF 5A C3 24 0F != Z R .!=!±c Z \$.
000004E0	3C 09 76 02 04 07 04 30	C3 51 8A E0 E8 EF FF 86 <.v....0 Qèαφn â
000004F0	C4 B1 04 D2 E8 E8 E6 FF	59 C3 53 8A FC E8 E9 FF -T±φμ Y Sèⁿφφ
00000500	88 25 4F 88 05 4F 8A C7	E8 DE FF 88 25 4F 88 05 è%0è.0è ± è%0è.
00000510	5B C3 51 52 32 E4 33 D2	B9 0A 00 F7 F1 80 CA 30 [ QR2Σ3T ...±C!0
00000520	88 14 4E 33 D2 3D 0A 00	73 F1 3C 00 74 04 0C 30 è.N3T=.s±<.t..0
00000530	88 04 5A 59 C3 52 50 55	56 57 50 BA 7F 00 B4 09 è.ZY RPUVWP α. .
00000540	CD 21 58 83 EC 07 8B EC	50 8D 76 02 C6 44 01 24 !=!Xâ∞.i∞Piv.†D.\$
00000550	E8 BF FF 8D 54 01 B4 09	CD 21 B4 02 B2 2E CD 21 φγ iT. .!=! .!.!=!
00000560	58 8A C4 8D 76 02 E8 A9	FF 8D 54 01 B4 09 CD 21 Xè-iv.φ- iT. .!=!
00000570	E8 57 FF BA 8D 00 B4 09	CD 21 8A C7 E8 6A FF 50 φW   i. .!=!è ±j P
00000580	8A D0 B4 02 CD 21 58 8A	D4 B4 02 CD 21 E8 3A FF è! .!=!Xè .!.!=!±:
00000590	BA 9A 00 B4 09 CD 21 8D	76 05 C6 44 01 24 8B FE   U. .!=!iv.†D.\$i•
000005A0	8B C1 E8 55 FF 8A C3 E8	3F FF 88 44 FB 88 64 FC i!±U è ±? èD√èdⁿ
000005B0	8B D5 B4 09 CD 21 E8 11	FF 83 C4 07 5F 5E 5D 58 i .!.!=!φ. â-_^]X
000005C0	5A C3 1E 33 C0 50 B8 20	00 8E D8 1E B8 00 F0 8E Z .3Lpγ .Ä+.γ.≡Ä
000005D0	D8 BB FE FF 8A 07 1F E8	D6 FE E8 F7 FE B8 00 30 †γ • è. .φ  •φ~•γ.0
000005E0	CD 21 E8 50 FF CB	!=!±P T

Рисунок 8 – Вид «хорошего» EXE файла

В «хорошем» файле EXE содержится информация для загрузчика, сегмент стека, сегмент данных и сегмент кода (3 сегмента вместо одного в «плохом» EXE). Если из исходного текста EXE-программы убрать сегмент



стека, то код будет располагаться с адреса 200h. Отличие от «плохого» EXE в том, что в «хорошем» не резервируется дополнительно 100h, которые в COM файле требовались для PSP, поэтому адреса начала кода отличаются на 100h + S, где S – размер стека.

Здесь сегмент стека начинается по адресу 0x200 от начала файла, данные – 0x400 и код – 0x4B0; все адреса будут на 20016 меньше, если их брать от начала образа программы. Это видно в файле карты памяти:

```
Start Stop Length Name Class
00000H 001FFH 00200H STACK
00200H 002AEH 000AFH DATA
002B0H 003E5H 00136H TEXT
```

## Загрузка COM модуля в основную память.

### 1. Какой формат загрузки COM модуля? С какого адреса располагается код?

Запуск файла .COM в отладчике TD.EXE представлен на рис 9.

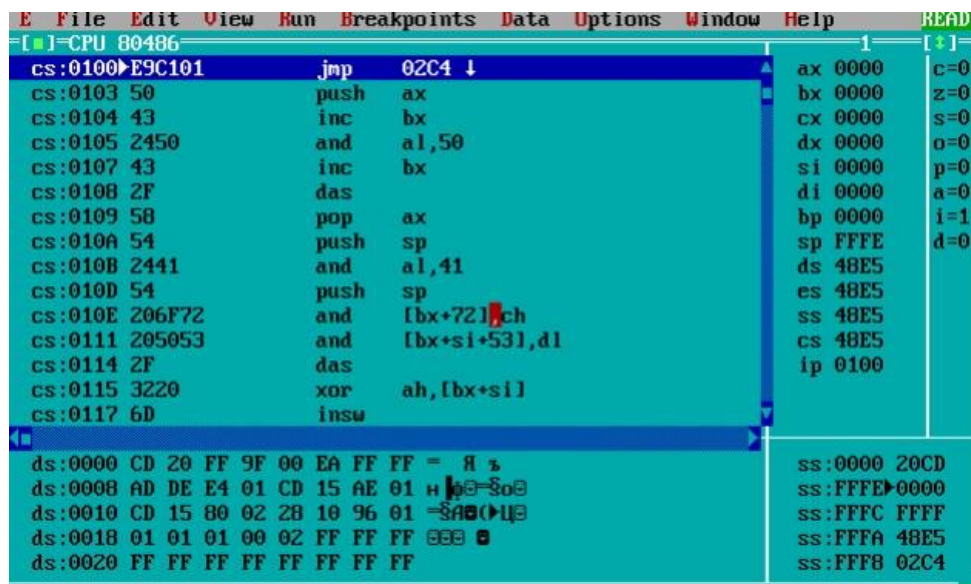


Рисунок 9 – Отладка файла .COM

Всё содержимое файла COM загружается в один и тот же сегмент по сдвигу 0x100; исполнение начинается с этого же адреса

### 2. Что располагается с 0 адреса?

PSP – Program Segment Prefix. Это структура, содержащая, в частности,

команду int 20h (завершение программы), объём доступной памяти в «параграфах», адреса подпрограмм, обрабатывающих некоторые события, 2 параметрические области и буфер передачи данных (DTA), который после запуска программы содержит остаток командной строки.

### 3. Какие значения имеют сегментные регистры? На какие области памяти они указывают?

Сегментные регистры имеют значения 48DDh и указывают на PSP.

### 4. Как определяется стек? Какую область памяти он занимает? Какие адреса?

В COM модуле нельзя объявить стек, он создается автоматически. Указатель стека устанавливается на конец сегмента – FFFEh. Стек занимает оставшуюся память, а его адреса изменяются от больших к меньшим, то есть от FFFEh к 0000h.

### Загрузка «хорошего» EXE модуля в основную память.

Запуск «хорошего» EXE модуля в отладчике TD.EXE представлен на рис.10.



Рисунок 10 – Загрузка «хорошего» EXE модуля в память

### 1. Как загружается «хороший» EXE? Какие значения имеют сегментные регистры?

В начале, модуль загружается в начальный сегмент. Потом,

обрабатывается таблица настройки, т.е. изменяются адреса в программе, которые зависят от адреса начального сегмента. 8 Сегментный регистр CS будет содержать адрес сегмента кода, регистр SS – адрес сегмента стека, а DS и ES – адрес сегмента, в котором по адресу 0 расположен PSP. Значение регистра DS, как правило, перезаписывается программой в начале выполнения адресом её сегмента данных. Перед этим его значение и слово, равное нулю добавляются в стек, чтобы из главной функции программы можно было выйти при помощи инструкции `ret far`. Это возможно, поскольку по адресу 0 в PSP лежит инструкция `int 20h`, которая завершает программу. В примере на скриншоте, образ программы загружен сразу за PSP. Начальный сегмент имеет адрес 0x48F5.

## **2. На что указывают регистры DS и ES?**

Они указывают на сегмент, в котором расположен PSP.

## **3. Как определяется стек?**

Стек может быть объявлен при помощи директивы `ASSUME`, которая устанавливает сегментный регистр SS на начало сегмента стека, а также задает значение SP, указанное в заголовке. Также стек может быть объявлен с помощью директивы `STACK`. Если стек не объявлять, то он будет создан автоматически.

## **4. Как определяется точка входа?**

Смещение точки входа в программу загружается в указатель команд IP и определяется операндом (функцией или меткой, с которой необходимо начать программу) директивы `END`.

## **Вывод.**

В ходе выполнения лабораторной работы были исследованы два формата загрузочных модулей в операционной системе DOS – COM и EXE (MZ). Исследованы различия в исходных текстах модулей, а также в их устройстве, формате и способе загрузки в память.

# ПРИЛОЖЕНИЕ А

## ИСХОДНЫЙ КОД ПРОГРАММЫ. LAB1COM.ASM

TEXT SEGMENT

ASSUME CS:TEXT, DS:TEXT

ORG 100h

START:

jmp MAIN

MODEL_FF	DB	'PC\$'
MODEL_FE_FB	DB	'PC/XT\$'
MODEL_FC	DB	'AT or PS/2 model 50 or 60\$'
MODEL_FA	DB	'PS/2 model 30\$'
MODEL_F8	DB	'PS/2 model 80\$'
MODEL_FD	DB	'PCjr\$'
MODEL_F9	DB	'PC Convertible\$'
MODEL_UNKNOWN	DB	'Unknown model\$'

MODEL\_TAB DW OFFSET MODEL\_FF

	DW	OFFSET	MODEL_FE_FB
	DW	OFFSET	MODEL_FE_FB
	DW	OFFSET	MODEL_FC
	DW	OFFSET	MODEL_FA
	DW	OFFSET	MODEL_F8
	DW	OFFSET	MODEL_FD
	DW	OFFSET	MODEL_F9
	DW	OFFSET	MODEL_UNKNOWN

MODEL\_CHARS DB 0FFh, 0FEh, 0FBh, 0FCh

DB 0FAh, 0F8h, 0FDh, 0F9h, 0

NEWLINE\_STRING DB 13, 10, '\$'

DOS\_VERSION\_STRING DB 'DOS version: \$'

```

OEM_NUMBER_STRING  DB    'OEM number: $'
USER_SERIAL_STRING DB    'User serial number: $'


LOOKUP_MODEL PROC

    push    BX
    xor BX,  BX

    ;; we don't save DX as we return into it
    anyway

    _lookup_model_loop:
        mov DL,  MODEL_CHARS[BX]

        ;; zero => unknown model
        test    DL,  DL
        jz  _lookup_model_break

        ;; model code matches => exit
        cmp DL,  AL
        jz  _lookup_model_break

        inc BX
        jmp _lookup_model_loop

_lookup_model_break:
    ;; we're addressing 2-byte words
    shl BX,  1
    mov DX,  MODEL_TAB[BX]

    pop BX
    ret
LOOKUP_MODEL ENDP


NEWLINE PROC

    push    DX

```

```

        mov DX,  OFFSET NEWLINE_STRING
        mov AH,  09h
        int 21h
        pop DX
        ret
NEWLINE ENDP

PRINT_MODEL PROC
        push     DX

        ;; assume the model name is in DX
        mov AH,  09h
        int 21h

        call     NEWLINE

        pop DX
        ret
PRINT_MODEL ENDP

TETR_TO_HEX PROC near
        and      AL, 0Fh
        cmp      AL, 09
        jbe      NEXT
        add      AL, 07
NEXT:
        add      AL, 30h
        ret
TETR_TO_HEX ENDP

BYTE_TO_HEX PROC near
        push     CX
        mov      AH, AL
        call     TETR_TO_HEX
        xchg     AL, AH

```



```

        mov     CL,4
        shr     AL,CL
        call    TETR_TO_HEX
        pop     CX
        ret
BYTE_TO_HEX ENDP

```

```

WRD_TO_HEX PROC  near
        push    BX
        mov     BH,AH
        call    BYTE_TO_HEX
        mov     [DI],AH
        dec     DI
        mov     [DI],AL
        dec     DI
        mov     AL,BH
        call    BYTE_TO_HEX
        mov     [DI],AH
        dec     DI
        mov     [DI],AL
        pop     BX
        ret
WRD_TO_HEX ENDP

```

```

BYTE_TO_DEC PROC  near
        push    CX
        push    DX
        xor     AH,AH
        xor     DX,DX
        mov     CX,10
loop_bd:
        div     CX
        or      DL,30h
        mov     [SI],DL
        dec     SI

```

```

                                xor     DX,DX
                                cmp     AX,10
                                jae     loop_bd
                                cmp     AL,00h
                                je      end_1
                                or      AL,30h
                                mov     [SI],AL
end_1:
                                pop     DX
                                pop     CX
                                ret
BYTE_TO_DEC    ENDP

```

```

PRINT_VERSION_INFO PROC

```

```

                                push    DX
                                push    AX
                                push    BP
                                push    SI
                                push    DI

```

```

                                push    AX

```

```

                                mov DX,  OFFSET DOS_VERSION_STRING
                                mov AH,  09h
                                int 21h

```

```

                                pop AX

```

```

;; reserver max buffer we're going to need
sub SP,  7
mov BP,  SP

```

```

;; print first line
push    AX

```

```

    lea SI,    [BP+2]
    mov BYTE PTR [SI+1],    '$'
    call      BYTE_TO_DEC

    lea DX,    [SI+1]
    mov  AH,    09h
    int  21h

    mov  AH,    02h
    mov  DL,    '.'
    int  21h

    pop  AX

    mov  AL,    AH
    lea SI,    [BP+2]
    call      BYTE_TO_DEC

    lea DX,    [SI+1]
    mov  AH,    09h
    int  21h

    call      NEWLINE

;; print second line
    mov  DX,    OFFSET OEM_NUMBER_STRING
    mov  AH,    09h
    int  21h

    mov  AL,    BH
    call      BYTE_TO_HEX

    push    AX
    mov  DL,    AL
    mov  AH,    02h

```

```

int 21h

pop AX
mov DL,  AH
mov AH,  02h
int 21h

call      NEWLINE

;; print third line
mov DX,   OFFSET USER_SERIAL_STRING
mov AH,   09h
int 21h

lea SI,   [BP+5]
mov BYTE PTR [SI+1], '$'
mov DI,   SI
mov AX,   CX
call      WRD_TO_HEX

mov AL,   BL
call      BYTE_TO_HEX
mov [SI-5], AL
mov [SI-4], AH

mov DX,   BP
mov AH,   09h
int 21h

call      NEWLINE

add SP,   7

pop DI
pop SI

```

```

        pop BP
        pop AX
        pop DX
        ret
PRINT_VERSION_INFO ENDP

MAIN PROC

        ;; AX <- 0F000:0FFFEh
        push    DS
        mov AX,  0F000h
        mov DS,  AX
        mov BX,  0FFFEh
        mov AL,  BYTE PTR [BX]
        pop DS

        ;; DX <- model name string offset (rel. to
DS)

        call    LOOKUP_MODEL

        call    PRINT_MODEL

        ;; AL == 00h => return OEM number
        ;; AL <- DOS version major
        ;; AH <- DOS version minor
        ;; BL:CX <- `user serial number'
        ;; BH <- MS-DOS OEM number
        mov AX,  3000h
        int 21h

        call    PRINT_VERSION_INFO

        mov AX,  4C00h
        int 21h

MAIN ENDP

```

TEXT ENDS

END START



## ПРИЛОЖЕНИЕ Б

### ИСХОДНЫЙ КОД ПРОГРАММЫ. LAB1EXE.ASM

```
STACK SEGMENT STACK
```

```
        DW 100h DUP(?)
```

```
STACK ENDS
```

```
        ASSUME CS:TEXT, DS:DATA, SS:STACK
```

```
DATA SEGMENT
```

```
MODEL_FF      DB  'PC$'
MODEL_FE_FB    DB  'PC/XT$'
MODEL_FC      DB  'AT or PS/2 model 50 or 60$'
MODEL_FA      DB  'PS/2 model 30$'
MODEL_F8      DB  'PS/2 model 80$'
MODEL_FD      DB  'PCjr$'
MODEL_F9      DB  'PC Convertible$'
MODEL_UNKNOWN DB  'Unknown model$'
```

```
MODEL_TAB DW OFFSET MODEL_FF
```

```
        DW OFFSET MODEL_FE_FB
        DW OFFSET MODEL_FE_FB
        DW OFFSET MODEL_FC
        DW OFFSET MODEL_FA
        DW OFFSET MODEL_F8
        DW OFFSET MODEL_FD
        DW  OFFSET MODEL_F9
        DW OFFSET MODEL_UNKNOWN
```

```
MODEL_CHARS    DB  0FFh, 0FEh, 0FBh, 0FCh
                DB  0FAh, 0F8h, 0FDh, 0F9h, 0
```

```
NEWLINE_STRING DB  13, 10, '$'
```

```

DOS_VERSION_STRING DB 'DOS version: $'
OEM_NUMBER_STRING DB 'OEM number: $'
USER_SERIAL_STRING DB 'User serial number: $'

DATA ENDS

TEXT SEGMENT

LOOKUP_MODEL PROC
    push    BX
    xor BX,  BX

    ;; we don't save DX as we return into it
    anyway

    _lookup_model_loop:
        mov DL,  MODEL_CHARS[BX]

        ;; zero => unknown model
        test    DL,  DL
        jz  _lookup_model_break

        ;; model code matches => exit
        cmp DL,  AL
        jz  _lookup_model_break

        inc BX
        jmp  _lookup_model_loop

    _lookup_model_break:
        ;; we're addressing 2-byte words
        shl BX,  1
        mov DX,  MODEL_TAB[BX]

        pop BX
        ret

```

```
LOOKUP_MODEL ENDP
```

```
NEWLINE PROC
```

```
    push    DX
    mov     DX, OFFSET NEWLINE_STRING
    mov     AH, 09h
    int     21h
    pop     DX
    ret
```

```
NEWLINE ENDP
```

```
PRINT_MODEL PROC
```

```
    push    DX

    ;; assume the model name is in DX
    mov     AH, 09h
    int     21h

    call    NEWLINE

    pop     DX
    ret
```

```
PRINT_MODEL ENDP
```

```
TETR_TO_HEX PROC near
```

```
    and     AL, 0Fh
    cmp     AL, 09
    jbe     NEXT
    add     AL, 07
```

```
NEXT:
```

```
    add     AL, 30h
    ret
```

```
TETR_TO_HEX ENDP
```

```
BYTE_TO_HEX PROC near
```

```

        push    CX
        mov     AH,AL
        call    TETR_TO_HEX
        xchg    AL,AH
        mov     CL,4
        shr     AL,CL
        call    TETR_TO_HEX
        pop     CX
        ret

```

```

BYTE_TO_HEX ENDP

```

```

WRD_TO_HEX PROC  near
        push    BX
        mov     BH,AH
        call    BYTE_TO_HEX
        mov     [DI],AH
        dec     DI
        mov     [DI],AL
        dec     DI
        mov     AL,BH
        call    BYTE_TO_HEX
        mov     [DI],AH
        dec     DI
        mov     [DI],AL
        pop     BX
        ret

```

```

WRD_TO_HEX ENDP

```

```

BYTE_TO_DEC PROC  near
        push    CX
        push    DX
        xor     AH,AH
        xor     DX,DX
        mov     CX,10

```

```

loop_bd:

```

```

                                div     CX
                                or      DL, 30h
                                mov     [SI], DL
                                dec     SI
                                xor     DX, DX
                                cmp     AX, 10
                                jae     loop_bd
                                cmp     AL, 00h
                                je      end_1
                                or      AL, 30h
                                mov     [SI], AL
end_1:
                                pop     DX
                                pop     CX
                                ret
BYTE_TO_DEC    ENDP

```

```

PRINT_VERSION_INFO PROC

```

```

    push     DX
    push     AX
    push     BP
    push     SI
    push     DI

```

```

    push     AX

```

```

    mov DX,  OFFSET DOS_VERSION_STRING
    mov AH,  09h
    int 21h

```

```

    pop AX

```

```

;; reserver max buffer we're going to need
sub SP,  7
mov BP,  SP

```

```

;; print first line
push     AX

lea SI,   [BP+2]
mov BYTE PTR [SI+1], '$'
call     BYTE_TO_DEC

lea DX,   [SI+1]
mov AH,   09h
int 21h

mov AH,   02h
mov DL,   '.'
int 21h

pop AX

mov AL,   AH
lea SI,   [BP+2]
call     BYTE_TO_DEC

lea DX,   [SI+1]
mov AH,   09h
int 21h

call     NEWLINE

;; print second line
mov DX,   OFFSET OEM_NUMBER_STRING
mov AH,   09h
int 21h

mov AL,   BH
call     BYTE_TO_HEX

```



```

push     AX
mov DL,  AL
mov AH,  02h
int 21h

pop AX
mov DL,  AH
mov AH,  02h
int 21h

call     NEWLINE

;; print third line
mov DX,  OFFSET USER_SERIAL_STRING
mov AH,  09h
int 21h

lea SI,  [BP+5]
mov BYTE PTR [SI+1],  '$'
mov DI,  SI
mov AX,  CX
call     WRD_TO_HEX

mov AL,  BL
call     BYTE_TO_HEX
mov [SI-5],  AL
mov [SI-4],  AH

mov DX,  BP
mov AH,  09h
int 21h

call     NEWLINE

```

```

        add SP, 7

        pop DI
        pop SI
        pop BP
        pop AX
        pop DX
        ret

PRINT_VERSION_INFO ENDP

MAIN PROC FAR

        push DS
        xor AX, AX
        push AX

        mov AX, DATA
        mov DS, AX

        ;; AX <- 0F000:0FFFEh
        push DS
        mov AX, 0F000h
        mov DS, AX
        mov BX, 0FFFEh
        mov AL, BYTE PTR [BX]
        pop DS

        ;; DX <- model name string offset (rel. to
DS)

        call LOOKUP_MODEL

        call PRINT_MODEL

        ;; AL == 00h => return OEM number
        ;; AL <- DOS version major
        ;; AH <- DOS version minor

```

```

        ;; BL: CX <- `user serial number'
        ;; BH <- MS-DOS OEM number
mov AX, 3000h
int 21h

        call PRINT_VERSION_INFO

        ret

MAIN ENDP

TEXT ENDS

END MAIN

```