Data-Coverage for Category-A Flight Software

Andoni Arregi, Fabian Schriever, Joan Roig

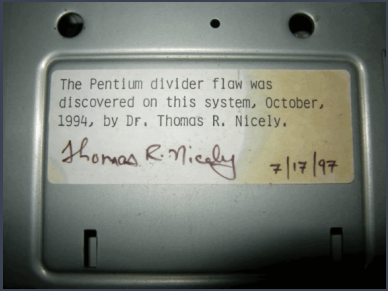
Content

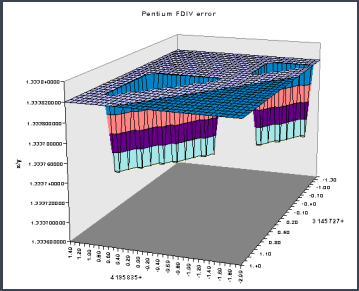
1. Motivation
2. Data in Flight Software
3. Tools
4. Conclusions

Motivation

Pentium FDIV Bug

Sometimes software bugs are related to **data**...





In fact, about 30% of the software bugs in aerospace software; 15% configurable data, 15% input data (Prokop, 2023)

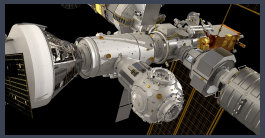
European Need for Category-A Software

Why does Europe need Category-A Qualified Software?

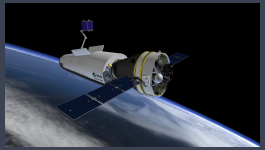
Because we had, have, and will have human-rated missions.

ATV

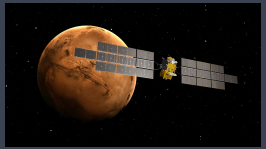
Orion ESM



I-Hab & ERM



Space-Rider



MRS – ERO

ADRIOS

ECSS Software Qualification Requirements

ECSS requirements are very **instruction oriented** but almost **nothing** is required **for data**.

Instruction vs Data

Although computer programs are made of instructions and data, they are not treated the same from a qualification point of view.

We just have more metrics and tools focused on instructions...

Data in Flight Software

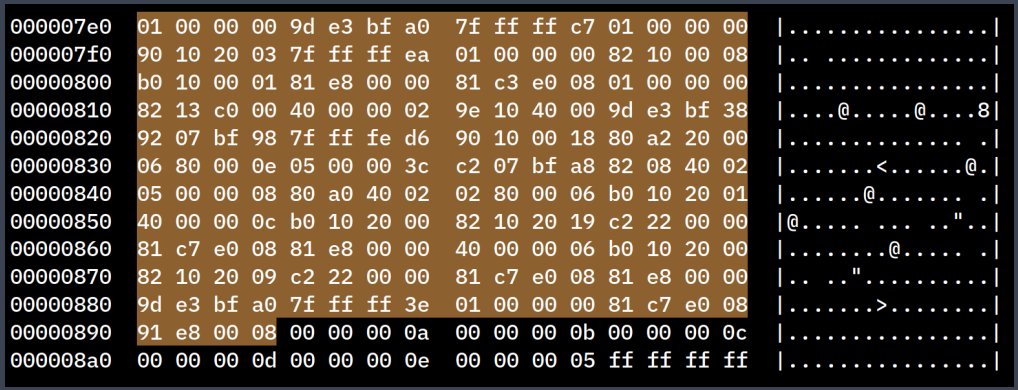
We have data **within** the flight software...

```
000007e0  01 00 00 00 9d e3 bf a0  7f ff ff c7 01 00 00 00  |................|
000007f0  90 10 20 03 7f ff ff ea  01 00 00 00 82 10 00 08  |.. .............|
00000800  b0 10 00 01 81 e8 00 00  81 c3 e0 08 01 00 00 00  |................|
00000810  82 13 c0 00 40 00 00 02  9e 10 40 00 9d e3 bf 38  |....@.....@....8|
00000820  92 07 bf 98 7f ff fe d6  90 10 00 18 80 a2 20 00  |.............. .|
00000830  06 80 00 0e 05 00 00 3c  c2 07 bf a8 82 08 40 02  |.......<......@.|
00000840  05 00 00 08 80 a0 40 02  02 80 00 06 b0 10 20 01  |......@....... .|
00000850  40 00 00 0c b0 10 20 00  82 10 20 19 c2 22 00 00  |@..... ... .."..|
00000860  81 c7 e0 08 81 e8 00 00  40 00 00 06 b0 10 20 00  |........@..... .|
00000870  82 10 20 09 c2 22 00 00  81 c7 e0 08 81 e8 00 00  |.. .."..........|
00000880  9d e3 bf a0 7f ff ff 3e  01 00 00 00 81 c7 e0 08  |.......>........|
00000890  91 e8 00 08 00 00 00 0a  00 00 00 0b 00 00 00 0c  |................|
000008a0  00 00 00 0d 00 00 00 0e  00 00 00 05 ff ff ff ff  |................|
```

Program object-code

Bits, Instructions, and Data

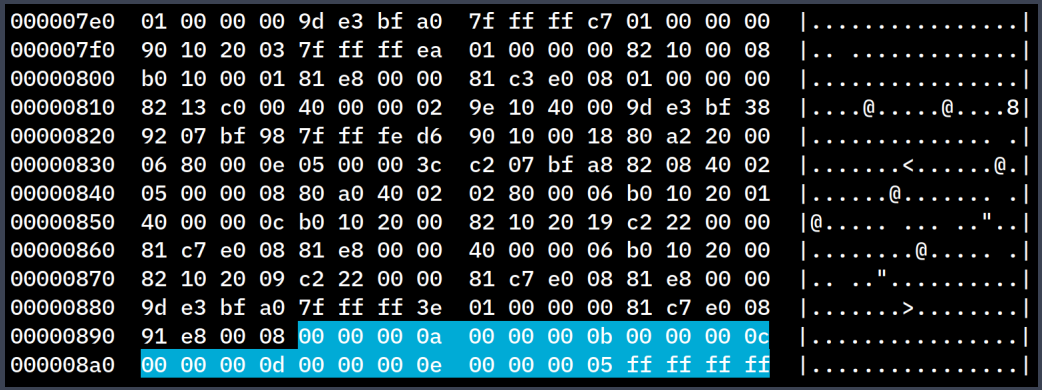
We have data **within** the flight software...



```
000007e0  01 00 00 00 9d e3 bf a0  7f ff ff c7 01 00 00 00  |................|
000007f0  90 10 20 03 7f ff ff ea  01 00 00 00 82 10 00 08  |.. .............|
00000800  b0 10 00 01 81 e8 00 00  81 c3 e0 08 01 00 00 00  |................|
00000810  82 13 c0 00 40 00 00 02  9e 10 40 00 9d e3 bf 38  |....@.....@....8|
00000820  92 07 bf 98 7f ff fe d6  90 10 00 18 80 a2 20 00  |.............. .|
00000830  06 80 00 0e 05 00 00 3c  c2 07 bf a8 82 08 40 02  |.......<......@.|
00000840  05 00 00 08 80 a0 40 02  02 80 00 06 b0 10 20 01  |......@....... .|
00000850  40 00 00 0c b0 10 20 00  82 10 20 19 c2 22 00 00  |@..... ... .."..|
00000860  81 c7 e0 08 81 e8 00 00  40 00 00 06 b0 10 20 00  |........@..... .|
00000870  82 10 20 09 c2 22 00 00  81 c7 e0 08 81 e8 00 00  |.. .."..........|
00000880  9d e3 bf a0 7f ff ff 3e  01 00 00 00 81 c7 e0 08  |.......>........|
00000890  91 e8 00 08 00 00 00 0a  00 00 00 0b 00 00 00 0c  |................|
000008a0  00 00 00 0d 00 00 00 0e  00 00 00 05 ff ff ff ff  |................|
```

Instructions within the object-code

We have data **within** the flight software...



```
000007e0  01 00 00 00 9d e3 bf a0  7f ff ff c7 01 00 00 00  |................|
000007f0  90 10 20 03 7f ff ff ea  01 00 00 00 82 10 00 08  |.. .............|
00000800  b0 10 00 01 81 e8 00 00  81 c3 e0 08 01 00 00 00  |................|
00000810  82 13 c0 00 40 00 00 02  9e 10 40 00 9d e3 bf 38  |....@.....@....8|
00000820  92 07 bf 98 7f ff fe d6  90 10 00 18 80 a2 20 00  |.............. .|
00000830  06 80 00 0e 05 00 00 3c  c2 07 bf a8 82 08 40 02  |.......<......@.|
00000840  05 00 00 08 80 a0 40 02  02 80 00 06 b0 10 20 01  |......@....... .|
00000850  40 00 00 0c b0 10 20 00  82 10 20 19 c2 22 00 00  |@..... ... .."..|
00000860  81 c7 e0 08 81 e8 00 00  40 00 00 06 b0 10 20 00  |........@..... .|
00000870  82 10 20 09 c2 22 00 00  81 c7 e0 08 81 e8 00 00  |.. .."..........|
00000880  9d e3 bf a0 7f ff ff 3e  01 00 00 00 81 c7 e0 08  |.......>........|
00000890  91 e8 00 08 00 00 00 0a  00 00 00 0b 00 00 00 0c  |................|
000008a0  00 00 00 0d 00 00 00 0e  00 00 00 05 ff ff ff ff  |................|
```

Data within the object-code

Other Data around Flight Software

We also have a lot of data **around** the flight software, affecting its behavior.

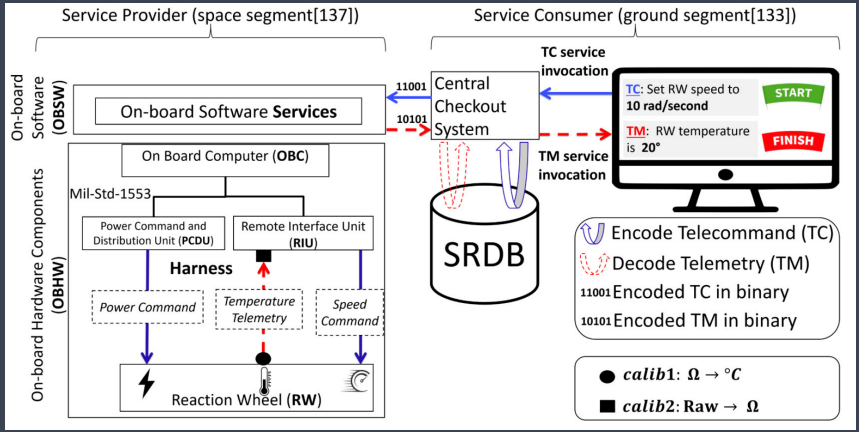


Figure taken from *Satellite Reference Databases scope and data organization: A literature review*, Malik Khalfallah; inspired by Olivier Notebaert

Tools

The Tools We Developed

Work carried out under ESA contract N. 4000143017/23/NL/AS/nh

The tools

- All open-source based (Alternatively proprietary tools can be used)
- Assist in the following tasks:
  - Assess data-coverage on a function unit-test basis
  - Show data-coverage projected on the source-code (as done by `gcov/lcov/gcovr`)
  - Assess data-usage of data-related memory areas/sections (on an executable basis)

The tools will be published as open source tools once completed at: https://gitlab.com/gtd-gmbh.

Assessing Data-Coverage of a Function

Where 20 unit-tests give complete statement and decision coverage:

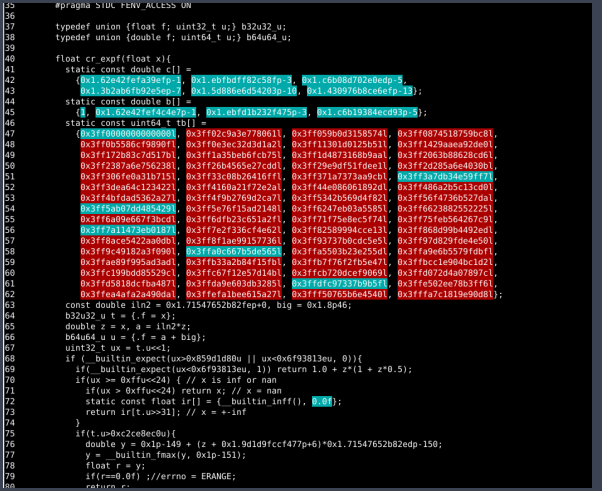
Filename	Line Coverage	Functions	Branches
example_w_exaf.c	100.0 % 54 / 54	100.0 % 3 / 3	94.4 % 17 / 18

```
56        :          :     0x3ff7a11473eb0187l, 0x3ff7e2f336cf4e62l, 0x3ff82589994cce13l, 0x3ff88d099b4492edl,
57        :          :     0x3ff8ace5422aa0dbl, 0x3ff8f1ae99157736l, 0x3ff97373b6dc5e5l, 0x3ff97d829fde4e50l,
58        :          :     0x3ff9c49182a3f090l, 0x3ffa0c667b5de565l, 0x3ffa6550b2bc255dl, 0x3ffa9e6b5579fdbfl,
59        :          :     0x3ffae89f995ad3adl, 0x3ffb33a2b84f15fbl, 0x3ffb7f76f2fb5e47l, 0x3ffbcc1e904bc1d2l,
60        :          :     0x3ffc199bdd85529cl, 0x3ffcc67f12e57d14bl, 0x3ffcb728dcef9869l, 0x3ffd0724a07897c1l,
61        :          :     0x3ffd5818dcfba487l, 0x3ffda9e603db3285l, 0x3ffdfc97337b9b5fl, 0x3ffe502ee78b3ff6l,
62        :          :     0x3ffea4afa2a490dal, 0x3ffeffa1beee615a27l, 0x3ffff5076b8e4548l, 0x3ffffa7c1819e90d8l};
63        :         20 :   const double ln2 = 0x1.71547652b82fep+0, big = 0x1.8p46;
64        :         20 :   b32u32_u t = {.f = x};
65        :         20 :   double z = x, a = ln2hf+z;
66        :         20 :   b64u64_u u = {.f = a + big};
67        :         20 :   uint32_t ux = t.u<<1;
68  [ + + + + ]:    20 :   if (__builtin_expect(ux>0x859d1d8bu || ux<0x6ff03813eu, 0)){
69  [ + + ]:        12 :     if(__builtin_expect(ux<0x6ff03813eu, 1)) return 1.0 + z*(1 + z*0.5);
70  [ + + ]:         6 :     if(ux >= 0xffeu<<24) { // x is inf or nan
71  [ + + ]:         3 :       if(ux > 0xffeu<<24) return x; // x = nan
72        :           :       static const float ir[] = {__builtin_inff(), 0.0f};
73        :         2 :       return ir[t.u>>31]; // x = +-inf
74        :           :     }
75  [ + + ]:         3 :     if(t.u>0xc2ce8ec0u){
76        :         2 :       double y = 0x1p-149 + (z + 0x1.9d1d9fccf477p+6)*0x1.71547652b82edp-150;
77        :         2 :       y = __builtin_fmax(y, 0x1p-151);
78        :         2 :       float r = y;
79        :           :       if(r==0.0f) ;//errno = ERANGE;
80        :         2 :       return r;
81        :           :     }
82  [ + ]:          1 :     if(t.u>0x42b17217u){
83        :         1 :       float r = 0x1p127f * 0x1p127f;
84        :           :       if(r>0x1.fffffep127f) ;//errno = ERANGE;
85        :         1 :       return r;
86        :           :     }
87        :           :   }
88        :         8 :   double ia = big - u.f, h = a + ia;
89        :         8 :   b64u64_u sv = {.u = tb[u.u&0x3f] + ((u.u>>6)<<52)};
90        :         8 :   double h2 = h*h, r = ((b[0] + h*b[1]) + h2*(b[2] + h*(b[3])))*sv.f;
91        :         8 :   float ub = r, lb = r - r*1.45e-10;
92  [ + + ]:        8 :   if(__builtin_expect(ub == lb, 0)){
93        :         1 :     const double ln2h = 0x1.7154765p+0, ln2l = 0x1.5c17f0bbbe88p-31;
94        :         1 :     double h = (ln2h*z + ia) + ln2l*z, s = sv.f, h2 = h*h, w = s*h;
95        :         1 :     double r = s + w*((c[0] + h*c[1]) + h2*((c[2] + h*c[3]) + h2*(c[4] + h*c[5])));
96        :         1 :     ub = r;
97        :           :   }
98        :         8 :   return ub;
99        :           : }
```

The missing coverage for one of the decisions is deactivated code in round to nearest

Assessing Data-Coverage of a Function

Data-coverage is by far not complete:



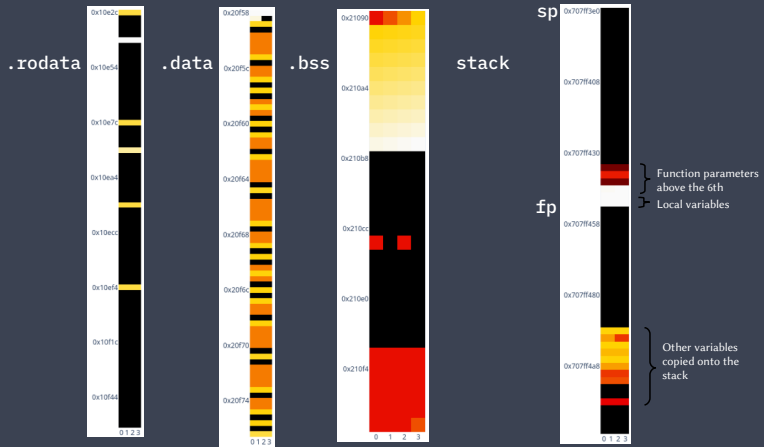
```
#pragma STDC FENV_ACCESS ON

typedef union {float f; uint32_t u;} b32u32_u;
typedef union {double f; uint64_t u;} b64u64_u;

float cr_expf(float x){
  static const double c[] =
    {0x1.62e42fefa39efp-1, 0x1.ebfbdff82c58fp-3, 0x1.c6b08d702e0edp-5,
     0x1.3b2ab6fb92e5ep-7, 0x1.5d886e6d54203p-10, 0x1.430976b8ce6efp-13};
  static const double b[] =
    {1, 0x1.62e42fef4c4e7p-1, 0x1.ebfd1b232f475p-3, 0x1.c6b19384ecd93p-5};
  static const uint64_t tb[] =
    {0x3ff0000000000000, 0x3ff02c9a3e778061, 0x3ff059b0d3158574, 0x3ff0874518759bc8,
     0x3ff0b5586cf9890f, 0x3ff0e3ec32d3d1a2, 0x3ff11301d0125b51, 0x3ff1429aaea92de0,
     0x3ff172b83c7d517b, 0x3ff1a35beb6fcb75, 0x3ff1d4873168b9aa, 0x3ff2063b88628cd6,
     0x3ff2387a6e756238, 0x3ff26b4565e27cdd, 0x3ff29e9df51fdee1, 0x3ff2d285a6e4030b,
     0x3ff306fe0a31b715, 0x3ff33c08b26416ff, 0x3ff371a7373aa9cb, 0x3ff3a7db34e59ff7,
     0x3ff3dea64c123422, 0x3ff4160a21f72e2a, 0x3ff44e086061892d, 0x3ff486a2b5c13cd0,
     0x3ff4bfdad5362a27, 0x3ff4f9b2769d2ca7, 0x3ff534286569d4f8, 0x3ff56f4736b527da,
     0x3ff5ab07dd485429, 0x3ff5e76f15ad2148, 0x3ff6247eb03a5585, 0x3ff66238825522251,
     0x3ff6a09e667f3bcd, 0x3ff6dfb23c651a2f, 0x3ff71f75e8ec5f74, 0x3ff75feb564267c9,
     0x3ff7a11473eb0187, 0x3ff7e2f336cf4e62, 0x3ff8258999ace131, 0x3ff868d99b4492ed,
     0x3ff8ace5422aa0db, 0x3ff8f1ae99157736, 0x3ff93737b0cdc5e5, 0x3ff97d829fde4e50,
     0x3ff9c49182a3f090, 0x3ff9fb0c667b5de65, 0x3ffa5503b23e255d, 0x3ffa9e6b5579fdbf1,
     0x3ffae89f995ad3ad, 0x3ffb33a2b84f15fb, 0x3ffb7f76f2fb5e47, 0x3ffbcc1e904bc1d2,
     0x3ffc199bdd85529c, 0x3ffc67f12e57d14b, 0x3ffcb720dcef90691, 0x3ffd072d4a07897cb,
     0x3ffd5818dcfba487, 0x3ffda9e603db3285, 0x3ffdfdc9733fb9b5f, 0x3ffe502ee78b3ff6f,
     0x3ffeaa4afa2a490da, 0x3ffefa1bee615a27, 0x3fff50765b6e4540, 0x3fffa7c1819e90d81};
  const double iln2 = 0x1.71547652b82fep+0, big = 0x1.8p46;
  b32u32_u t = {.f = x};
  double z = x, a = iln2*z;
  b64u64_u u = {.f = a + big};
  uint32_t ux = t.u<<1;
  if (__builtin_expect(ux>0x859d1d80u || ux<0x6f93813eu, 0)){
    if(__builtin_expect(ux<0x6f93813eu, 1)) return 1.0 + z*(1 + z*0.5);
    if(ux >= 0xffu<<24) { // x is inf or nan
      if(ux > 0xffu<<24) return x; // x = nan
      static const float ir[] = {__builtin_inff(), 0.0f};
      return ir[t.u>>31]; // x = +-inf
    }
    if(t.u>0xc2ce8ec0u){
      double y = 0x1p-149 + (z + 0x1.9d1d9fccf477p+6)*0x1.71547652b82edp-150;
      y = __builtin_fmax(y, 0x1p-151);
      float r = y;
      if(r==0.0f) ;//errno = ERANGE;
      return r;
```

Assessing Data Usage on a Whole Executable

The tools enable the logging and visualization of data accesses to memory sections and areas of interest:



Conclusions

Conclusions

Lack of Data-Coverage Requirements

- Data-coverage is as meaningful for Cat-A software as instruction coverage.
- We have a lack of data verification and coverage requirements in ECSS.
- Data elements affecting flight-software behavior are not properly validated for Cat-A software.

Method and Tools for Data-Coverage

- We propose some methods and tools to ensure that data affecting the flight software behavior gets exercised by tests.

Backup

Assessing Data-Coverage of a Function

This is an extract of a implementation of the `expf()` function (`core-math` library):

```
float cr_expf(float x){
    static const double c[] = {0x1.62e42fefa39efp-1, 0x1.ebfbdff82c58fp-3, 0x1.c6b08d702e0edp-5,
                               0x1.3b2ab6fb92e5ep-7, 0x1.5d886e6d54203p-10, 0x1.430976b8ce6efp-13};
    static const double b[] = {1, 0x1.62e42fef4c4e7p-1, 0x1.ebfd1b232f475p-3, 0x1.c6b19384ecd93p-5};
    static const uint64_t tb[] = {0x3ff0000000000000l, 0x3ff02c9a3e778061l, 0x3ff059b0d3158574l, 0x3ff0874518759bc8l,
                                  0x3ff0b5586cf9890fl, 0x3ff0e3ec32d3d1a2l, 0x3ff11301d0125b51l, 0x3ff1429aaea92de0l,
                                  0x3ff172b83c7d517bl, 0x3ff1a35beb6fcb75l, 0x3ff1d4873168b9aal, 0x3ff2063b88628cd6l,
                                  0x3ff2387a6e756238l, 0x3ff26b4565e27cddl, 0x3ff29e9df51fdee1l, 0x3ff2d285a6e4030bl,
                                  ...
                                  0x3ffae89f995ad3adl, 0x3ffb33a2b84f15fbl, 0x3ffb7f76f2fb5e47l, 0x3ffbcc1e904bc1d2l,
                                  0x3ffc199bdd85529cl, 0x3ffc67f12e57d14bl, 0x3ffcb720dcef9069l, 0x3ffd072d4a07897cl,
                                  0x3ffd5818dcfba487l, 0x3ffda9e603db3285l, 0x3ffdfc97337b9b5fl, 0x3ffe502ee78b3ff6l,
                                  0x3ffea4afa2a490dal, 0x3ffefa1bee615a27l, 0x3fff50765b6e4540l, 0x3fffa7c1819e90d8l};
    const double iln2 = 0x1.71547652b82fep+0, big = 0x1.8p46;
    b32u32_u t = {.f = x};
    ...
    uint32_t ux = t.u<<1;
    if (__builtin_expect(ux>0x859d1d80u || ux<0x6f93813eu, 0)){
            ...
    }
```

Assessing Data Usage on a Whole Executable


```
//
// DATA

const uint32_t constant_rodata[74] = { 10, 11, 12, 13, 14, 15, 16, 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 }; // in .rodata

const uint8_t const_size = 74;          // in .rodata
const uint8_t not_init_size = 120;      // in .rodata
static char uninitialized_data[120];    // in .bss
uint32_t size1 = 40;                    // in .data
uint16_t size2 = 24;                    // in .data
uint32_t not_constant_data[74] = { 10, 11, 12, 13, 14, 15, 16, 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 }; // in .data

//
// FUNCTIONS

int32_t init();
int32_t foo1(uint32_t index1, uint16_t index2, uint32_t index3,
             uint32_t index4, uint8_t index5, uint32_t index6);
int32_t foo2(uint32_t tmp1, uint32_t tmp2, uint32_t tmp3, uint32_t tmp4,
             uint32_t tmp5, uint32_t tmp6, uint32_t tmp7, uint32_t tmp8);
```