Meltdown and Spectre Samples

Written in Assembly

U. Plonus
u.plonus@gmail.com

April 2, 2018

Contents

1 Introduction **5**
- 1.1 Overview . . . 5
- 1.2 Conventions . . . 5
  - 1.2.1 Introduction . . . 5
  - 1.2.2 Data Sections . . . 5
- 1.3 Nasm . . . 5

2 Cache Access Timing **7**
- 2.1 Introduction . . . 7
- 2.2 Detect Cache Access Time . . . 7
  - 2.2.1 High Resolution Timer . . . 7
  - 2.2.2 Cache Access Time Routine . . . 7
- 2.3 Measure Cache Access Time . . . 8
  - 2.3.1 Setup . . . 8
  - 2.3.2 Measure Time . . . 9
- 2.4 Read Byte via Cache Access Time . . . 12
  - 2.4.1 Introduction . . . 12
  - 2.4.2 Clear Cache for Measurement . . . 12
  - 2.4.3 Indexed Array Access . . . 13
  - 2.4.4 Read a Byte from the Cache . . . 14
  - 2.4.5 The Whole Program to Read a Byte from Cache . . . 16
  - 2.4.6 Improve Cache Access Time Analysis . . . 19
- 2.5 Read Array via Cache Access Time . . . 24
  - 2.5.1 Introduction . . . 24
  - 2.5.2 Setup . . . 24
  - 2.5.3 Reading Bytes via Cache . . . 25
  - 2.5.4 Printing the Results . . . 30

3 Signals **37**
- 3.1 Basics . . . 37
- 3.2 Detecting Signals . . . 37
- 3.3 Handling Signals . . . 37

4 Utilities **39**
- 4.1 Introduction . . . 39
- 4.2 Common Chunks . . . 39
  - 4.2.1 Exit Program . . . 39

4.2.2 Stack Frame 39
4.3 Random Number Generator 40
4.4 Printing Strings 41
4.4.1 Printing Strings with Length 41
4.4.2 Printing C-Strings 41
4.5 Printing Numbers 42
4.5.1 Printing a Decimal 64bit Unsigned Integer 42
4.5.2 Printing a Hexadecimal 8bit Integer 45

A Glossary **47**

B Acronyms **49**

C x86-Instructions **51**

D Code Chunks **53**

E License **55**
E.1 GNU Free Documentation License 55
E.2 Code License 64
E.2.1 GNU GENERAL PUBLIC LICENSE 64
E.2.2 Code Chunk of GPL 78

1 Introduction

1.1 Overview

TBD

1.2 Conventions

1.2.1 Introduction

In this section we define some convention that are specific for this document.

1.2.2 Data Sections

The data is divided into three parts: read-only data, initialized data and uninitialized data. Code chunks with this type of data will all have defined sufficies.

Definition 1 *Read-only data is data that is not modified during program execution. The suffix for read-only data is **-rodata**.*

Definition 2 *Initialized data is data that is changeable during program execution. The data is already initialized with data when the program starts. The suffix for initialized data is **-idata**.*

Definition 3 *Uninitialized data is data that is changeable during program execution. The data is not initialized. The suffix for uninitialized data is **-udata**.*

1.3 Nasm

TBD

5 ⟨*preamble* 5⟩≡ (11f 18c 21a 23c 34b)

```
bits 64

⟨license 78⟩

        global          _start
        pagesize        equ 4096
```

Defines:
`_start`, used in chunks 9a, 14a, 21b, and 24.
`pagesize`, used in chunks 8b, 9c, 13, 14c, 16c, 22c, 24, 25, 28, 29, and 34b.

2 Cache Access Timing

2.1 Introduction

TBD

2.2 Detect Cache Access Time

2.2.1 High Resolution Timer

First we need a high resolution timer to determine the cache access time. For this we use the time stamp counter. The time stamp counter is monotonically incrementing. When reading the time stamp counter (with `rdtsc`) the result is delivered back in the registers `EDX` and `EAX` forming a 64bit value. The time stamp counter is not an absolute value but a relative value, meaning that you cannot (easily) calculate from the time stamp counter to some time units (e.g. ns). But this is no problem as we only want to measure relative times.

To retrieve a 64bit value for the time we shift the value in `EDX` 32 bits to the left and add the value of `EAX` to this.

7 ⟨*tsc-64bit* 7⟩≡ (8a)

```
rdtsc
shl     RDX,32
add     RAX,RDX
```

2.2.2 Cache Access Time Routine

Next we need a routine that calculates the cache access time for us.

First we have to ensure in this routine that the speculative execution of the processor does not interfere with our time measurement. For this we use the instruction `lfence` which ensures that all previous reads are done before executing the next instructions.

Next we access a memory location with the address `RDI` by loading this into `RCX` and measure the time before and after the access.

The command `lfence` before reading the time stamp counter is needed because we have to ensure that all reads before the time measurements are done.

At last we calculate the relative time needed to access the memory location. In theory we should see a difference whether the memory location is accessed before or not.

Parameters

RDI the address of the memory which is loaded either from the cache or from memory

Return

RAX the relative time of the cache access

8a ⟨*calculate-cache-access-time* 8a⟩≡ (11f 18c 21a 23c 34b)
```
_calccachetime:
      lfence
  ⟨tsc-64bit 7⟩
      mov       R8,RAX
      mov       RCX,[RDI]
      lfence
  ⟨tsc-64bit 7⟩
      sub       RAX,R8
      ret
```
Defines:
`_calccachetime`, used in chunks 10b, 11b, 15b, and 26b.

2.3 Measure Cache Access Time

2.3.1 Setup

To measure the cache timing we create a standalone program that shows us the time for a cached and for an uncached memory access.

First we need some area in memory with data which we can later read from. This data area goes into the area `.bss` which contains uninitialized data. We align the data at a page boundary and reserve one pages for our data.

8b ⟨*data-udata* 8b⟩≡ (11f 18c 21a 23c 34b)
```
      alignb        pagesize
      data:         resb pagesize
```
Defines:
`data`, used in chunks 9, 11b, 14b, 17a, 18a, 22–25, 29a, and 34.
Uses `pagesize` 5.

From time to time we need a small scratch area so we define an area with 32 bytes.

8c ⟨*scratch-udata* 8c⟩≡ (11f 18c 21a 23c)
```
      scratch:      resb 32
```
Defines:
`scratch`, used in chunks 10f, 11d, 17, 18a, 23b, 26b, and 30–32.

The program begins with the label `_start`.

9a ⟨*cachetiming-program* 9a⟩≡ (11f) 9f▷

```
_start:
```

Uses `_start` 5.

Now we start with initialising the `data` area with some random data. For this we load `RDI` with the address of the `data` area.

9b ⟨*init-random-data* 9b⟩≡ (9f 14a 21b) 9c▷

```
        mov         RDI,data
```

Uses `data` 8b.

Next we load the number of bytes to fill into `RSI`. For this we load the `pagesize` into `RSI`.

9c ⟨*init-random-data* 9b⟩+≡ (9f 14a 21b) ◁9b 9d▷

```
        mov         RSI,pagesize
```

Uses `pagesize` 5.

At last we load `EDX` with some random seed. For this we use `rdtsc` and only use the lower 32 bit of the value.

9d ⟨*init-random-data* 9b⟩+≡ (9f 14a 21b) ◁9c 9e▷

```
        rdtsc
        mov         EDX,EAX
```

Now we call `_xorshift` to fill the `data` area.

9e ⟨*init-random-data* 9b⟩+≡ (9f 14a 21b) ◁9d

```
        call        _xorshift
```

Uses `_xorshift` 40a.

Now we add this `data` initialization to our program.

9f ⟨*cachetiming-program* 9a⟩+≡ (11f) ◁9a 9g▷

```
  ⟨init-random-data 9b⟩
```

2.3.2 Measure Time

Now that we have setup our `data` area we can now cache data from the first page by loading it into a register which also loads this into the cache.

For this we load `RDI` with the address of the `data` area.

9g ⟨*cachetiming-program* 9a⟩+≡ (11f) ◁9f 10a▷

```
        mov         RDI,data
```

Uses `data` 8b.

Before we load the data into a register now we will clear the cache lines with the given address. For this we use the instruction `clflush`. After flushing the cache line we ensure (with `lfence`) that all reads from memory are finished before we load the data into a register again (and filling the cache).

10a ⟨*cachetiming-program* 9a⟩+≡ (11f) ◁9g 10b▷

```
clflush   [RDI]
lfence
mov       RCX,[RDI]
```

Now we can determine the time that is needed to load this data once again. We do not need to load `RDI` again because it has not changed.

10b ⟨*cachetiming-program* 9a⟩+≡ (11f) ◁10a 10e▷

```
call      _calccachetime
```

Uses `_calccachetime` 8a.

Now we have the relative cache access time in register `RAX`. We store this value to the stack and print out an explaining text.

For this we define the text to print.

10c ⟨*cachetiming-rodata* 10c⟩≡ (11f) 11c▷

⟨*common-rodata* 10d⟩

```
scached:        db "Cached Access Time: ",0x00
```

Defines:
`scached`, used in chunk 10e.

Additionally we define some helper data, in this case line feed (LF).

10d ⟨*common-rodata* 10d⟩≡ (10c 18c 21a 23c 30a)

```
slf:            db 0x0a
```

Defines:
`slf`, used in chunks 11, 17, 18a, 23b, 26b, and 33a.

Now we can store `RAX` and print the text.

10e ⟨*cachetiming-program* 9a⟩+≡ (11f) ◁10b 10f▷

```
push      RAX
mov       RDI,scached
call      _print
```

Uses `_print` 42a and `scached` 10c.

We now restore the value and print the measured time to `stdout`.

10f ⟨*cachetiming-program* 9a⟩+≡ (11f) ◁10e 11a▷

```
pop       RDI
mov       RSI,scratch
call      _printdu64bit
```

Uses `_printdu64bit` 43a and `scratch` 8c.

At last we append a LF to the output.

11a ⟨*cachetiming-program* 9a⟩+≡ (11f) ◁10f 11b▷

```
mov       RSI,slf
mov       RDI,1
call      _nprint
```

Uses `_nprint` 41b and `slf` 10d.

Now we do the same with an uncached value. The difference is that we do not load the value before.

11b ⟨*cachetiming-program* 9a⟩+≡ (11f) ◁11a 11d▷

```
mov       RDI,data
clflush   [RDI]
lfence
call      _calccachetime
```

Uses `_calccachetime` 8a and `data` 8b.

Now we have the time of the uncached data access in `RAX` and can print it out with some explaining text.

11c ⟨*cachetiming-rodata* 10c⟩+≡ (11f) ◁10c

```
suncached:     db "Uncached Access Time: ",0x00
```

Defines:
`suncached`, used in chunk 11d.

11d ⟨*cachetiming-program* 9a⟩+≡ (11f) ◁11b 11e▷

```
push      RAX
mov       RDI,suncached
call      _print
pop       RDI
mov       RSI,scratch
call      _printdu64bit
mov       RSI,slf
mov       RDI,1
call      _nprint
```

Uses `_nprint` 41b, `_print` 42a, `_printdu64bit` 43a, `scratch` 8c, `slf` 10d, and `suncached` 11c.

At last we exit the program.

11e ⟨*cachetiming-program* 9a⟩+≡ (11f) ◁11d

⟨*exitProgram* 39b⟩

Now we can put everything together and have our `cachetiming` program that we can now execute.

11f ⟨*cachetiming.asm* 11f⟩≡

⟨*preamble* 5⟩

```
section .rodata
⟨cachetiming-rodata 10c⟩

section .bss
⟨data-udata 8b⟩
⟨scratch-udata 8c⟩

section .text
⟨cachetiming-program 9a⟩

⟨calculate-cache-access-time 8a⟩

⟨xorshift-prng 40a⟩

⟨utilities 39a⟩
```

The program source is placed in `asm/`. With `make` in the folder we can create an executable which is moved to `bin/`. There we can execute this program.

```
$ bin/cachetiming
Cached Access Time: 72
Uncached Access Time: 372
$
```

2.4 Read Byte via Cache Access Time

2.4.1 Introduction

We have seen that we can determine if the content of a memory address is in the cache or not (see 2.3 Measure Cache Access Time).

So next we try to read a single byte from the memory by only detecting the cache access time.

2.4.2 Clear Cache for Measurement

Before we can determine the cache access times we need to clear the cache. We define a subroutine for this.

Parameters

RDI	the address of the probe memory
RSI	the step size in the probe memory

13a ⟨*clearcache* 13a⟩≡ (18c 21a 23c)
```
  _clearcache:
       mov      RCX,256
       cld
  .nextflush:
       clflush  [RDI]
       add      RDI,RSI
       loop     .nextflush
       lfence
       ret
```
Defines:
_clearcache, used in chunks 13b, 22a, and 28c.

Now we add this to our program.

13b ⟨*cachereadbyte-program* 13b⟩≡ (18c 21a) 14a ▷
```
       mov      RDI,probe
       mov      RSI,pagesize
       call     _clearcache
```
Uses _clearcache 13a 26a, pagesize 5, and probe 13c 25a.

2.4.3 Indexed Array Access

To read the value of a byte via the cache we use the byte to index into a probe array and then determine the cache access times of this probe array.

For this we will first create a `probe` array.

13c ⟨*probe-udata* 13c⟩≡ (18c 21a 23c)
```
       alignb         pagesize
       probe          times 256 resb pagesize
```
Defines:
probe, used in chunks 13, 14c, 16c, 22c, 25, 28, and 29.
Uses pagesize 5.

Next we will fill this `probe` array with some random data (similar to the chunks for `data` 9b, 9c, 9d and 9e).

13d ⟨*init-random-probe* 13d⟩≡ (14a 21b)
```
       mov      RDI,probe
       mov      RSI,pagesize
       shl      RSI,8
       rdtsc
       mov      EDX,EAX
       call     _xorshift
```
Uses _xorshift 40a, pagesize 5, and probe 13c 25a.

Now we add the initialization of the data and probe area to the program.

14a ⟨*cachereadbyte-program* 13b⟩+≡ (18c 21a) ◁13b 14b▷
```
_start:
⟨init-random-data 9b⟩
⟨init-random-probe 13d⟩
```
Uses _start 5.

Now we can read a byte from data into AL.

14b ⟨*cachereadbyte-program* 13b⟩+≡ (18c 21a) ◁14a 14c▷
```
        mov     RDI,data
        xor     RAX,RAX
        mov     AL,[RDI]
```
Uses data 8b.

We use the value in RAX to access the probe array.

14c ⟨*cachereadbyte-program* 13b⟩+≡ (18c 21a) ◁14b 16c▷
```
        mov     RDX,pagesize
        mul     RDX
        mov     RSI,probe
        mov     RAX,[RSI+RAX]
```
Uses pagesize 5 and probe 13c 25a.

Now we read the datum back via the cache access times. For this we create subroutines.

2.4.4 Read a Byte from the Cache

First we create a subroutine to read the cache access timings for the probe area.

Parameters

RDI	the address of the probe memory
RSI	the step size in the probe memory
RDX	an area to keep the detected cache access times (256 * 8 bytes)

14d ⟨*readcachetiming* 14d⟩≡ (18c 21a 23c) 15a▷
```
_readcachetiming:
⟨enterstackframe 39c⟩
```
Defines:
_readcachetiming, used in chunks 16c and 22c.

Now we create space on the stack to keep the variables. Next we save the parameters to the stack space created.

15a ⟨*readcachetiming* 14d⟩+≡ (18c 21a 23c) ◁14d 15b▷

```
      sub       RSP,32
      mov       [RBP-8],RDI
      mov       [RBP-16],RSI
      mov       [RBP-24],RDX
```

Now we can start detecting the cache access times.

15b ⟨*readcachetiming* 14d⟩+≡ (18c 21a 23c) ◁15a 15c▷

```
      mov       RCX,256
  .nextcacheread:
      mov       [RBP-32],RCX
      call      _calccachetime
      mov       RDX,[RBP-24]
      mov       [RDX],RAX
      add       RDX,8
      mov       [RBP-24],RDX
      mov       RDI,[RBP-8]
      add       RDI,[RBP-16]
      mov       [RBP-8],RDI
      mov       RCX,[RBP-32]
      loop      .nextcacheread
```

Uses `_calccachetime` 8a.

At the end we clean up the stack again and return to the caller.

15c ⟨*readcachetiming* 14d⟩+≡ (18c 21a 23c) ◁15b

```
  ⟨leavestackframe 39d⟩
      ret
```

After we determined all cache access times we can now find the lowest access time and with this the possible byte. We return two results from this subroutine, in `AL` the byte with the lowest cache access time and in `AH` the count of the lowest cache access time. Only if `AH` is 1 then the value in `AL` is valid.

Parameters

`RDI` the area with the detected cache access times (256 * 8 bytes)

Return

`AL` the read byte (in `AL`) with the lowest cache access time

`AH` the number of bytes read with the lowest cache access time

16a ⟨*analyzecachemintiming* 16a⟩≡ (18c)

```
 _analyzecachetiming:
      push    RDI
      mov     R8,0xffffffffffffff
      xor     R9,R9
      xor     RCX,RCX
      mov     RSI,RDI
 .nexttry:
      lodsq
      cmp     RAX,R8
      ja      .nohit
      mov     R8,RAX
      mov     R9,RCX
 .nohit:
      inc     RCX
      cmp     RCX,256
      jb      .nexttry
      xor     RCX,RCX
      pop     RSI
 .nextcount:
      lodsq
      cmp     RAX,R8
      ja      .nomin
      inc     R10
 .nomin:
      inc     RCX
      cmp     RCX,256
      jb      .nextcount
      mov     RAX,R10
      shl     RAX,8
      mov     AL,R9b
      ret
```

2.4.5 The Whole Program to Read a Byte from Cache

Before we can start using our new subroutine `_readcachetiming` we need to define a data area for the cache access times.

16b ⟨*timings-udata* 16b⟩≡ (18c 21a 23c)

```
      timings        resq 256
```

Now we have all subroutines together we now can start implementing the main program and output the byte read.

16c ⟨*cachereadbyte-program* 13b⟩+≡ (18c 21a) ◁14c 17b▷

```
        mov         RDI,probe
        mov         RSI,pagesize
        mov         RDX,timings
        call        _readcachetiming
        mov         RDI,timings
        call        _analyzecachetiming
```

Uses _readcachetiming 14d, pagesize 5, and probe 13c 25a.

Now we define a string to output for the read byte and the expected byte.

17a ⟨*cachereadbyte-rodata* 17a⟩≡ (18c 21a 23c)

```
        sreadbyte:     db "Byte read via cache access:     ",0x00
        ssountbyte:    db "Count of bytes with min timing: ",0x00
        sexpectedbyte: db "Expected byte from data:        ",0x00
```

Uses data 8b.

We save the value from RAX (only AL is interesting to us) to the stack and print out the text.

17b ⟨*cachereadbyte-program* 13b⟩+≡ (18c 21a) ◁16c 17c▷

```
        push        RAX
        mov         RDI,sreadbyte
        call        _print
```

Uses _print 42a.

Now we print the read byte and end the line with a LF.

17c ⟨*cachereadbyte-program* 13b⟩+≡ (18c 21a) ◁17b 17d▷

```
        pop         RDI
        push        RDI
        and         RDI,0xff
        mov         RSI,scratch
        call        _printh8bit
        mov         RDI,1
        mov         RSI,slf
        call        _nprint
```

Uses _nprint 41b, _printh8bit 45b, scratch 8c, and slf 10d.

Next we print (for information) the number of bytes read with the minimum cache access timing.

17d ⟨*cachereadbyte-program* 13b⟩+≡ (18c 21a) ◁17c 18a▷

```
        mov         RDI,ssountbyte
        call        _print
        pop         RDI
        shr         RDI,8
        and         RDI,0xff
        mov         RSI,scratch
```

```
        call      _printdu64bit
        mov       RDI,1
        mov       RSI,slf
        call      _nprint
```

Uses _nprint 41b, _print 42a, _printdu64bit 43a, scratch 8c, and slf 10d.

Now we read the byte from the original data array and print this also.

18a ⟨*cachereadbyte-program* 13b⟩+≡ (18c 21a) ◁17d 18b▷

```
        mov       RDI,sexpectedbyte
        call      _print
        mov       RSI,data
        xor       RAX,RAX
        mov       AL,[RSI]
        mov       RDI,RAX
        mov       RSI,scratch
        call      _printh8bit
        mov       RDI,1
        mov       RSI,slf
        call      _nprint
```

Uses _nprint 41b, _print 42a, _printh8bit 45b, data 8b, scratch 8c, and slf 10d.

At last we exit the program.

18b ⟨*cachereadbyte-program* 13b⟩+≡ (18c 21a) ◁18a

⟨*exitProgram* 39b⟩

Now we put all together to get the program `cachereadbyte` that we can execute.

18c ⟨*cachereadbyte.asm* 18c⟩≡

```
⟨preamble 5⟩

section .rodata
⟨common-rodata 10d⟩
⟨cachereadbyte-rodata 17a⟩

section .bss
⟨data-udata 8b⟩
⟨probe-udata 13c⟩
⟨scratch-udata 8c⟩
⟨timings-udata 16b⟩

section .text
⟨cachereadbyte-program 13b⟩

⟨clearcache 13a⟩
```

⟨*calculate-cache-access-time* 8a⟩

⟨*readcachetiming* 14d⟩

⟨*analyzecachemintiming* 16a⟩

⟨*xorshift-prng* 40a⟩

⟨*utilities* 39a⟩

2.4.6 Improve Cache Access Time Analysis

As we can see – when running the program `cachereadbyte` – the result is not always as clear as it could be. Simply getting the lowest cache access time seems not to be enough.

Sample outputs of the program are

```
$ bin/cachereadbyte
Byte read via cache access:     2b
Count of bytes with min timing: 1
Expected byte from data:        2b
$ bin/cachereadbyte
Byte read via cache access:     ff
Count of bytes with min timing: 11
Expected byte from data:        b3
$ bin/cachereadbyte
Byte read via cache access:     2f
Count of bytes with min timing: 1
Expected byte from data:        87
$
```

So we have to improve our cache time detection routine. We will change the implementation of the chunk 16a to define a thrshold that is a little bit above the min access time and run the cache detection routine multiple times if no clear result is returned.

First start with the subrotuine to analyze the cache access timing. We define a threshold 25 % above the minimum cache access time.

First we search for the minimum cache access time.

Parameters

`RDI` the area with the detected cache access times (256 * 8 bytes)

Return

`AL` the first byte (in `AL`) with a cache access time below the threshold

`AH` the number of bytes read with a cache access time below the threshold

20a ⟨*analyzecachesimpthrestiming* 20a⟩≡ (21a 23c) 20b ▷

```
_analyzecachetiming:
        push    RDI
        mov     R8,0xffffffffffffffff
        xor     RCX,RCX
        mov     RSI,RDI
.nextmin:
        lodsq
        cmp     RAX,R8
        ja      .nonewmin
        mov     R8,RAX
.nonewmin:
        inc     RCX
        cmp     RCX,256
        jb      .nextmin
```

Now we have the minimum cache access time in R8. Next we will add $\frac{1}{4}$ to this to have our threshold.

20b ⟨*analyzecachesimpthrestiming* 20a⟩+≡ (21a 23c) ◁ 20a 20c ▷

```
        mov     RAX,R8
        shr     RAX,4
        add     R8,RAX
```

Now we scan the cache access times a second time and take all values below the threshold into account.

20c ⟨*analyzecachesimpthrestiming* 20a⟩+≡ (21a 23c) ◁ 20b

```
        pop     RSI
        xor     RCX,RCX
        xor     R9,R9
.nextbyte:
        lodsq
        cmp     RAX,R8
        ja      .nonewbyte
        inc     R9
        mov     R10,RCX
.nonewbyte:
        inc     RCX
        cmp     RCX,256
        jb      .nextbyte
        mov     RAX,R9
        shl     RAX,8
        mov     AL,R10b
        ret
```

Now we put all together to get the program `cachereadbyte2` that we can execute.

21a ⟨*cachereadbyte2.asm* 21a⟩≡

```
⟨preamble 5⟩

section .rodata
⟨common-rodata 10d⟩
⟨cachereadbyte-rodata 17a⟩

section .bss
⟨data-udata 8b⟩
⟨probe-udata 13c⟩
⟨scratch-udata 8c⟩
⟨timings-udata 16b⟩

section .text
⟨cachereadbyte-program 13b⟩

⟨clearcache 13a⟩

⟨calculate-cache-access-time 8a⟩

⟨readcachetiming 14d⟩

⟨analyzecachesimpthrestiming 20a⟩

⟨xorshift-prng 40a⟩

⟨utilities 39a⟩
```

Now when we only find a single hit then the possibility that the byte from the cache timing is the original byte is much higher.

Next we will create a program that tries to read the value from the cache until we have a single result.

First we initialize our data and probe areas.

21b ⟨*cachereadbyte3-program* 21b⟩≡ (23c) 22c ▷

```
_start:
⟨init-random-data 9b⟩
⟨init-random-probe 13d⟩
```

Uses `_start` 5.

Next we create a subroutine that clears the cache and reads in a byte via the probe array.

Parameters

RDI	the address of the byte to read
RSI	the address of the probe memory
RDX	the step size in the probe memory

22a ⟨*readbyte2cache* 22a⟩≡ (23c) 22b▷

```
_readbyte2cache:
      push    RDI
      push    RSI
      push    RDX
      mov     RDI,RSI
      mov     RSI,RDX
      call    _clearcache
```

Defines:
`_readbyte2cache`, used in chunk 22c.
Uses `_clearcache` 13a 26a.

Next we can add the read of the byte and caching the data from the probe array.

22b ⟨*readbyte2cache* 22a⟩+≡ (23c) ◁22a

```
      pop     RDX
      pop     RSI
      pop     RDI
      xor     RAX,RAX
      mov     AL,[RDI]
      mul     RDX
      mov     AL,[RSI+RAX]
      ret
```

Now we add the call to this subroutine to our program and determine the byte by analyzing the cache access times.

22c ⟨*cachereadbyte3-program* 21b⟩+≡ (23c) ◁21b 23a▷

```
.startreadcache:
      mov     RDI,data
      mov     RSI,probe
      mov     RDX,pagesize
      call    _readbyte2cache
      mov     RDI,probe
      mov     RSI,pagesize
      mov     RDX,timings
      call    _readcachetiming
      mov     RDI,timings
      call    _analyzecachetiming
```

Uses `_readbyte2cache` 22a, `_readcachetiming` 14d, `data` 8b, `pagesize` 5, and `probe` 13c 25a.

Now we check if the read byte was a single byte, else we will do this again.

23a ⟨*cachereadbyte3-program* 21b⟩+≡ (23c) ◁22c 23b▷

```
cmp     AH,1
ja      .startreadcache
```

Now we print out our result.

23b ⟨*cachereadbyte3-program* 21b⟩+≡ (23c) ◁23a

```
push    RAX
mov     RDI,sreadbyte
call    _print
pop     RDI
and     RDI,0xff
mov     RSI,scratch
call    _printh8bit
mov     RDI,1
mov     RSI,slf
call    _nprint
mov     RDI,sexpectedbyte
call    _print
mov     RSI,data
xor     RAX,RAX
mov     AL,[RSI]
mov     RDI,RAX
mov     RSI,scratch
call    _printh8bit
mov     RDI,1
mov     RSI,slf
call    _nprint
```

⟨*exitProgram* 39b⟩

Uses `_nprint` 41b, `_print` 42a, `_printh8bit` 45b, `data` 8b, `scratch` 8c, and `slf` 10d.

23c ⟨*cachereadbyte3.asm* 23c⟩≡

⟨*preamble* 5⟩

```
section .rodata
```

⟨*common-rodata* 10d⟩
⟨*cachereadbyte-rodata* 17a⟩

```
section .bss
```

⟨*data-udata* 8b⟩
⟨*probe-udata* 13c⟩
⟨*scratch-udata* 8c⟩
⟨*timings-udata* 16b⟩

```
section .text
⟨cachereadbyte3-program 21b⟩

⟨readbyte2cache 22a⟩

⟨clearcache 13a⟩

⟨calculate-cache-access-time 8a⟩

⟨readcachetiming 14d⟩

⟨analyzecachesimpthrestiming 20a⟩

⟨xorshift-prng 40a⟩

⟨utilities 39a⟩
```

Even if this program is not perfect because it is not reliable all the time it is reliable enough to demonstrate the next steps.

2.5 Read Array via Cache Access Time

2.5.1 Introduction

Now that we have seen that we can determine if a value was in the cache or not (see 2.3 Measure Cache Access Time) we will read a complete array of data by only measuring the cache access time.

2.5.2 Setup

For this we start with some `data` area that we can read later as defined before.

So start with the program and fill the `data` area with some random data.

24 ⟨*cacheread-program* 24⟩≡ (34b) 25b ▷

```
_start:
        mov     RDI,data
        mov     RSI,pagesize
        rdtsc
        mov     EDX,EAX
        call    _xorshift
```

Uses `_start` 5, `_xorshift` 40a, `data` 8b, and `pagesize` 5.

Next we will create a probe area that is $256 * pagesize$. We only access the first byte of each page but we divide the data over such a large area (1 MiB) to ensure that the cache lines that we use do not interfere each other.

25a ⟨*cacheread-udata* 25a⟩≡ (34b) 28a ▷

```
probe:          times 256 resb pagesize
```

Defines:
`probe`, used in chunks 13, 14c, 16c, 22c, 25, 28, and 29.
Uses `pagesize` 5.

Next we fill this area also with some random data.

25b ⟨*cacheread-program* 24⟩+≡ (34b) ◁ 24 29f ▷

```
mov         RDI,probe
mov         RAX,pagesize
mov         RCX,256
mul         RCX
mov         RSI,RCX
rdtsc
mov         EDX,EAX
call        _xorshift
```

Uses `_xorshift` 40a, `pagesize` 5, and `probe` 13c 25a.

2.5.3 Reading Bytes via Cache

As we saw we can determine if a memory datum is in cache or not. For reading a complete byte we have to do a little bit more. Basically we use the byte accessed to index a different `probe` area. Because the memory is not cached byte by byte but in so called cache lines we cannot use a simple 256 bytes sized probe array but must at least have a space between the accessed bytes that is larger than a cache line size. This is the reason why we use a probe array of 256 * `pagesize` bytes of size.

Basically we use the following code to access the data. We load the content of the address we want to probe into a register. Then we multiply the register with some arbitary value (we use `pagesize`) and the access the probe area with the calculated offset. We can then test the cache which page was cached and have our value from the `data`.

25c ⟨*cacheread-sample* 25c⟩≡

```
mov         RAX,[data]
mul         RAX,pagesize
mov         RBX,[probe+RAX]
```

Uses `data` 8b, `pagesize` 5, and `probe` 13c 25a.

First we write a subroutine to clear the cache lines from data from our `probe` area. We assume that we use 256 values (0 . . . 255) for the indexing into the probe array. Also the `probe` area must be at least 256 * RSI bytes in size.

Parameters

RDI the address of the probe array

RSI the interval between the probe addresses used

26a ⟨*clear-cache* 26a⟩≡ (34b)

```
_clearcache:
      cld
      mov       RCX,256
      xor       RAX,RAX
.clear_next:
      clflush   [RDI+RAX]
      add       RAX,RSI
      loop      .clear_next
      lfence
      ret
```

Defines:
`_clearcache`, used in chunks 13b, 22a, and 28c.

Next we need a subroutine that determines the cache line access times for the data in the `probe` area. So we create a subroutine that loops similar to the `_clearcache` subroutine over all addresses and measures the cache access time for each page.

Parameters

RDI the address of the probe array

RSI the interval between the probe addresses used

RDX the address of the results of the cache measurements. The area needs to be 256 * 8 bytes in size

26b ⟨*detect-cache-area-time* 26b⟩≡ (34b)

```
_calcareacachetime:
      xor       RCX,RCX
.next_timing:
      push      RCX
      push      RDX
      push      RDI
      push      RSI
      call      _calccachetime
      push      RAX
      mov       RDI,RAX
      mov       RSI,scratch
      call      _printdu64bit
      mov       RDI,1
```

```
        mov     RSI,slf
        call    _nprint
        pop     RAX
        pop     RSI
        pop     RDI
        pop     RDX
        pop     RCX
        mov     [RDX+8*RCX],RAX
        add     RDI,RSI
        inc     RCX
        cmp     RCX,256
        jb      .next_timing
        ret
```

Defines:
  `_calcareacachetime`, used in chunk 27.
Uses `_calccachetime` 8a, `_nprint` 41b, `_printdu64bit` 43a, `scratch` 8c, and `slf` 10d.

We now can determine the cache line with the lowest access time. This is the cache line that was cached before.

Parameters

`RDI`	the address of the probe array
`RSI`	the interval between the probe addresses used
`RDX`	the address of the results of the cache measurements. The area needs to be 256 * 8 bytes in size

Return

`RAX`	the byte (in `AL`) which is found by cache timing analysis

TBD

27 ⟨*detect-byte* 27⟩≡ (34b)

```
  _detectbytebycl:
        push    RDI
        call    _calcareacachetime
        pop     RDI
        mov     RSI,RDX
        xor     RCX,RCX
        mov     R8,0xffffffffffffffff
        xor     R9,R9
  .nextbyte:
        mov     RAX,[RDI+8*RCX]
        cmp     RAX,R8
```

```
        jb        .foundbyte
        inc       RCX
        cmp       RCX,256
        jae       .done
        jmp       .nextbyte
  .foundbyte:
        mov       R8,RAX
        mov       R9,RCX
        jmp       .nextbyte
  .done:
        mov       RAX,R9
        ret
```

Uses _calcareacachetime 26b.

Now we need some area to store all the data. Once we use an area for the timing data and another area for the read memory data.

28a ⟨*cacheread-udata* 25a⟩+≡ (34b) ◁25a 30c▷

```
        result:         resb pagesize
        timing:         resq 256
```

Uses pagesize 5.

Now we have the base for reading a complete memory area via a cache covert channel.
We now create a subroutine to loop over the memory we want to read and read the values back via the cache access time.
First we create a area where we can store the read bytes.

28b ⟨*readback-udata* 28b⟩≡ (34b)

```
  ;     readback:       align pagesize, resb pagesize
```

Defines:
  readback, never used.
Uses pagesize 5.

Now we create the subroutine that reads the bytes from the source array `data` and writes the results from the cache access time into `readback`.
First we setup a counter in `R8` and clear the cache.

28c ⟨*cache-readback* 28c⟩≡ (34b) 29a▷

```
  _cachereadback:
        xor       R8,R8
  .nextbyte:
        push      R8
        mov       RDI,probe
        mov       RSI,pagesize
        call      _clearcache
        pop       R8
```

Defines:
  _cachereadback, used in chunk 29f.
Uses _clearcache 13a 26a, pagesize 5, and probe 13c 25a.

Next we read in the data from the array.

29a ⟨*cache-readback* 28c⟩+≡ (34b) ◁28c 29b▷
```
        mov     RSI,data
        xor     RAX,RAX
        mov     AL,[RSI+R8]
```
Uses data 8b.

Next we use the read byte to index into our probe array.

29b ⟨*cache-readback* 28c⟩+≡ (34b) ◁29a 29c▷
```
        mov     RDX,pagesize
        mul     RDX
        mov     RSI,probe
        mov     AL,[RSI+RAX]
```
Uses pagesize 5 and probe 13c 25a.

Now we have put data into the cache that depends on the value read from `data`. Next we will read the cache access times to determine the data read.

29c ⟨*cache-readback* 28c⟩+≡ (34b) ◁29b 29d▷
```
        mov     RDI,probe
        mov     RSI,pagesize
        mov     RDX,timing
        push    R8
        call    _detectbytebycl
        pop     R8
```
Uses pagesize 5 and probe 13c 25a.

Next we store the read byte into our result array.

29d ⟨*cache-readback* 28c⟩+≡ (34b) ◁29c 29e▷
```
        mov     RDI,result
        mov     [RDI+R8],AL
```

Now we can increment our counter and check if there are more bytes to read. If no more bytes need to be read we leave our subroutine.

29e ⟨*cache-readback* 28c⟩+≡ (34b) ◁29d
```
        inc     R8
        cmp     R8,pagesize
        jb      .nextbyte
        ret
```
Uses pagesize 5.

After all we can now call this new subroutine and read our data by detecting the cache access times.

29f ⟨*cacheread-program* 24⟩+≡ (34b) ◁25b 34a▷
```
        call    _cachereadback
```
Uses _cachereadback 28c.

2.5.4 Printing the Results

Now we want to see the results so we now read a byte from the origin (`data`) and from our read back data (`readback`).

First we define some helpful data for colorizing the output.

30a ⟨*cacheread-rodata* 30a⟩≡ (34b)

```
⟨common-rodata 10d⟩
      sbgred:        db 0x1b,"[1;41m",0x00
      sresetstyle:   db 0x1b,"[0m",0x00
      sseparator:    db "- ",0x00
      sblank:        db " "
      semptybyte:    db "   ",0x00
```

Defines:
`sbgred`, used in chunk 32c.
`sblank`, used in chunks 31b and 32c.
`sresetstyle`, used in chunks 32c and 33a.
`sseparator`, used in chunk 32a.

First we define a subroutine which prints out up to 16 bytes each side by side on the screen. If two bytes in the arrays are different then the value at the right side (from the second array) will be printed with read background.

Parameters

`RDI`	the address of the first array
`RSI`	the address of the second array
`RDX`	number of bytes to print (up to 16). If the value is above 16 then nothing is printed out

30b ⟨*print-comparision16* 30b⟩≡ (34b) 30d ▷

```
  _printcompare16:
```

Defines:
`_printcompare16`, used in chunk 33b.

Additionally we need some scratch area for the printing.

30c ⟨*cacheread-udata* 25a⟩+≡ (34b) ◁ 28a

```
      scratch:       resb 64
```

Uses `scratch` 8c.

At the start of the subroutine we prepare a stack frame for further operations as we will need to save and restore the registers `RDI`, `RSI`, `RDX` and `RCX` multiple times. Additionally we store `R12` to the stack to use this register as scratch register.

30d ⟨*print-comparision16* 30b⟩+≡ (34b) ◁ 30b 31a ▷

```
      push     RBP
      mov      RBP,RSP
```

```
        sub         RSP,32
        mov         [RBP-8],RDI
        mov         [RBP-16],RSI
        mov         [RBP-24],RDX
        push        R12
```

Now we first start and check that no more than 16 bytes should be printed, otherwise we will end the subroutine immediately.

31a ⟨*print-comparision16* 30b⟩+≡ (34b) ◁30d 31b▷

```
        cmp         RDX,0x10
        ja          .done
```

Next we can start and handle the ”left” side of the output. We output up to 16 bytes and then continue at `.leftbytesdone` (31c).

31b ⟨*print-comparision16* 30b⟩+≡ (34b) ◁31a 31c▷

```
        xor         RCX,RCX
  .nextbyteleft:
        cmp         RCX,RDX
        mov         [RBP-32],RCX
        jae         .leftbytesdone
        mov         AL,[RDI+RCX]
        xor         AH,AH
        mov         DI,AX
        mov         RSI,scratch
        call        _printh8bit
        mov         RDI,1
        mov         RSI,sblank
        call        _nprint
        mov         RDI,[RBP-8]
        mov         RDX,[RBP-24]
        mov         RCX,[RBP-32]
        inc         RCX
        jmp         .nextbyteleft
```

Uses `_nprint` 41b, `_printh8bit` 45b, `sblank` 30a, and `scratch` 8c.

Now we fill up the space so that the space of 16 bytes is occupied.

31c ⟨*print-comparision16* 30b⟩+≡ (34b) ◁31b 32a▷

```
  .leftbytesdone:
        cmp         RCX,0x10
        jae         .leftdone
        mov         RDI,semptybyte
        call        _print
        inc         RCX
        jmp         .leftbytesdone
```

```
  .leftdone:
```

Uses _print 42a.

Next we print out the separator between the two compare block.

32a ⟨*print-comparision16* 30b⟩+≡ (34b) ◁31c 32b▷

```
        mov         RDI,sseparator
        call        _print
```

Uses _print 42a and sseparator 30a.

To print the second half (for comparision) we restore the values of the parameters first.

32b ⟨*print-comparision16* 30b⟩+≡ (34b) ◁32a 32c▷

```
        mov         RDI,[RBP-8]
        mov         RSI,[RBP-16]
        mov         RDX,[RBP-24]
```

Now we compare each byte with the original value first and then print it out. If the value differs from the original value we additionally mark the byte.

32c ⟨*print-comparision16* 30b⟩+≡ (34b) ◁32b 33a▷

```
        xor         RCX,RCX
  .nextbyteright:
        mov         [RBP-32],RCX
        cmp         RCX,RDX
        jae         .rightbytesdone
        mov         AL,[RSI+RCX]
        mov         AH,[RDI+RCX]
        mov         R12W,AX
        cmp         AH,AL
        je          .printplain
        mov         RDI,sbgred
        call        _print
  .printplain:
        xor         RDI,RDI
        mov         AX,R12W
        xor         AH,AH
        mov         DI,AX
        mov         RSI,scratch
        call        _printh8bit
        mov         AX,R12W
        cmp         AH,AL
        je          .printdone
        mov         RDI,sresetstyle
        call        _print
  .printdone:
```

```
        mov        RDI,1
        mov        RSI,sblank
        call       _nprint
        mov        RDI,[RBP-8]
        mov        RSI,[RBP-16]
        mov        RDX,[RBP-24]
        mov        RCX,[RBP-32]
        inc        RCX
        jmp        .nextbyteright
  .rightbytesdone:
```

Uses `_nprint` 41b, `_print` 42a, `_printh8bit` 45b, `sbgred` 30a, `sblank` 30a, `scratch` 8c, and `sresetstyle` 30a.

33a ⟨*print-comparision16* 30b⟩+≡ (34b) ◁32c

```
        cmp        RCX,0x10
        jae        .rightdone
        inc        RCX
        jmp        .rightbytesdone
  .rightdone:
  .done:
        mov        RDI,sresetstyle
        call       _print
        mov        RDI,1
        mov        RSI,slf
        call       _nprint
        pop        R12
        mov        RSP,RBP
        pop        RBP
        ret
```

Uses `_nprint` 41b, `_print` 42a, `slf` 10d, and `sresetstyle` 30a.

TBD

Parameters

`RDI` the address of the first array

`RSI` the address of the second array

`RDX` number of bytes to print. In each line 16 bytes from the first and 16 bytes from the right side are printed

33b ⟨*print-comparision* 33b⟩≡ (34b)

```
  _printcompare:
        mov        RDX,16
        call       _printcompare16
```

```
        ret
```

Defines:
  `_printcompare`, used in chunk 34a.
Uses `_printcompare16` 30b.

TBD

34a ⟨*cacheread-program* 24⟩+≡ (34b) ◁29f

```
        mov     RDI,data
        mov     RSI,result
        call    _printcompare
```

Uses `_printcompare` 33b and `data` 8b.

34b ⟨*cacheread.asm* 34b⟩≡

```
  ⟨preamble 5⟩

  section .bss
        align           pagesize
  ⟨data-udata 8b⟩
  ⟨cacheread-udata 25a⟩
  ⟨readback-udata 28b⟩

  section .data
  ⟨cacheread-rodata 30a⟩

  section .text
  ⟨cacheread-program 24⟩

  ⟨exitProgram 39b⟩

  ⟨print-comparision 33b⟩

  ⟨print-comparision16 30b⟩

  ⟨cache-readback 28c⟩

  ⟨clear-cache 26a⟩

  ⟨calculate-cache-access-time 8a⟩

  ⟨detect-cache-area-time 26b⟩

  ⟨detect-byte 27⟩

  ⟨xorshift-prng 40a⟩
```

$\langle$*utilities* 39a$\rangle$

Uses `data` 8b and `pagesize` 5.

3 Signals

3.1 Basics

TBD

3.2 Detecting Signals

TBD

3.3 Handling Signals

TBD

4 Utilities

4.1 Introduction

TBD

39a ⟨*utilities* 39a⟩≡ (11f 18c 21a 23c 34b)

⟨*nprint* 41b⟩

⟨*print* 42a⟩

⟨*printdu64bit* 43a⟩

⟨*printh8bit* 45b⟩

4.2 Common Chunks

4.2.1 Exit Program

This chunk ends the program with exit code 0.

39b ⟨*exitProgram* 39b⟩≡ (11e 18b 23b 34b)

```
        xor     RDI,RDI
        mov     RAX,60
        syscall
```

4.2.2 Stack Frame

A chunk to create a stack frame.

39c ⟨*enterstackframe* 39c⟩≡ (14d)

```
        push    RBP
        mov     RBP,RSP
```

A chunk to clean up the created stack frame.

39d ⟨*leavestackframe* 39d⟩≡ (15c)

```
        mov     RSP,RBP
        pop     RBP
```

4.3 Random Number Generator

To initialize the data a random number generator (RNG) is used. The sample programs use xorshift[1] as RNG.

First we clear the direction flag to ensure that we are incrementing the data pointer `RDI`.

Next we move the number of values to be generated to `RCX` (which is a counter in x86 processors) and divide it by 4 (because we use a 32bit RNG). Additionally we move the seed to `EAX`.

Parameters

`RDI`	the address of the memory which is to be filled with random numbers
`RSI`	the number of bytes that are filled with random numbers. This must be a multiple of 4
`EDX`	the seed of the RNG

40a ⟨*xorshift-prng* 40a⟩≡ (11f 18c 21a 23c 34b) 40b ▷

```
_xorshift:
        cld
        mov         RCX,RSI
        shr         RCX,2
        mov         EAX,EDX
```

Defines:
`_xorshift`, used in chunks 9e, 13d, 24, and 25b.

Now we can generate the next 32bit random number.

40b ⟨*xorshift-prng* 40a⟩+≡ (11f 18c 21a 23c 34b) ◁ 40a 41a ▷

```
.next_random:
        mov         EBX,EAX
        shl         EAX,13
        xor         EAX,EBX
        mov         EBX,EAX
        shr         EAX,17
        xor         EAX,EBX
        mov         EBX,EAX
        shl         EAX,5
        xor         EAX,EBX
```

[1] https://en.wikipedia.org/wiki/Xorshift

Because we want to generate multiple random numbers we store the value of EAX to [RDI] and loop for the next random number.

41a ⟨*xorshift-prng* 40a⟩+≡ (11f 18c 21a 23c 34b) ◁40b

```
        stosd
        loop      .next_random
        ret
```

4.4 Printing Strings

4.4.1 Printing Strings with Length

The routine _nprint prints a string with the given length to stdout.

We move the number of bytes to print to RDX which is the 3rd parameter to the systemcall. Next we move the address of the bytes to print to RSI which is the 2nd parameter to the systemcall. The 1st argument (in RDI) to the systemcall is the file descriptor (1 is stdout). Additionally the number of the systemcall (1) is passed in RAX. The systemcall (syscall) now prints RDX bytes from [RSI] to the file descriptor RDI.

At the end we return to the caller.

Parameters

RDI the number of bytes to print to stdout

RSI the address to the bytes to print to stdout

41b ⟨*nprint* 41b⟩≡ (39a)

```
  _nprint:
        mov       RDX,RDI
        mov       RDI,1
        mov       RAX,1
        syscall
        ret
```

Defines:
_nprint, used in chunks 11, 17, 18a, 23b, 26b, 31–33, 42d, 45a, and 46a.

4.4.2 Printing C-Strings

The routine _print prints a null-terminated string to stdout.

First we clear the direction flag to increment the address in RDI while scanning the data.

Next we start with clearing AL (setting it to null) and saving the address of the string to RSI. We're using RSI because we later need the address to calculate the length of the string.

Parameters

RDI the address to the null-terminated bytes to print to stdout

42a ⟨*print* 42a⟩≡ (39a) 42b▷

```
_print:
    cld
    xor     AL,AL
    mov     RSI,RDI
```

Defines:
_print, used in chunks 10e, 11d, 17, 18a, 23b, and 31–33.

Next we search for the terminating null ('\0') character. For this we use the instruction scasb (scan string byte) which compares the byte at the address [RDI] with the value in AL and sets the flags accordingly. When the byte at [RDI] is not the value of AL the next instruction (jne) jumps to the given label (.next_char in this case).

scasb additionally increments RDI so that we go through the string until '\0' is found.

42b ⟨*print* 42a⟩+≡ (39a) ◁42a 42c▷

```
.next_char:
    scasb
    jne     .next_char
```

After we have found the string termination we calculate the number of bytes that the string has. In RSI we now have the starting address of the bytes to print and in RDI we have the end address of the bytes to print. After that we calculate the number of bytes to print.

42c ⟨*print* 42a⟩+≡ (39a) ◁42b 42d▷

```
    sub     RDI,RSI
```

Now we have the address of the string in RDI and the length of the string in RSI which are the 1st and 2nd argument in the call of _nprint.

42d ⟨*print* 42a⟩+≡ (39a) ◁42c

```
    call    _nprint
    ret
```

Uses _nprint 41b.

4.5 Printing Numbers

4.5.1 Printing a Decimal 64bit Unsigned Integer

The routine _printdu64bit prints a given 64bit integer as unsigned decimal number to stdout.

To print a decimal number we have to divide the number by 10 and get the remainder for printing (from right to left). For this we move the divisor to a register and the dividend to RAX. We have to use RAX because this is the only register we can use for division.

Additionally we need the address of the scratch area in RDI for storing the result. We also save the address of the scratch area to R8 for later use.

To increment the address during the processing we clear the direction flag.

Parameters

RDI the number number to print to `stdout`

RSI the address of a scratch area with a size of at least 20 bytes

43a ⟨*printdu64bit* 43a⟩≡ (39a) 43b ▷

```
_printdu64bit:
    mov     RAX,RDI
    mov     RDI,RSI
    mov     R8,RDI
    mov     RCX,10
    cld
```

Defines:
`_printdu64bit`, used in chunks 10f, 11d, 17d, and 26b.

Now we define a label to jump back when we see that there are still more digits to print. Then we test RAX for 0 and end the processing of the digits.

43b ⟨*printdu64bit* 43a⟩+≡ (39a) ◁ 43a 43c ▷

```
.next:
    cmp     RAX,0
    je      .done
```

Next we divide RAX by RCX. For this we have to clear RDX because this is the higher value of the dividend. The result is then placed into RAX and the remainder into RDX.

43c ⟨*printdu64bit* 43a⟩+≡ (39a) ◁ 43b 43d ▷

```
    xor     RDX,RDX
    div     RCX
```

We now exchange the result and the remainder because we now need the remainder in RAX (or AL) for further processing. Now we can add the ASCII character '0' to AL and have the correct ASCII value in AL. Now we can store the ASCII character to the scratch area.

43d ⟨*printdu64bit* 43a⟩+≡ (39a) ◁ 43c 44a ▷

```
    xchg    RDX,RAX
    add     AL,'0'
    stosb
```

Now we restore RAX (which we saved to RDX) to go into the next round.

44a ⟨*printdu64bit* 43a⟩+≡ (39a) ◁43d 44b▷

```
        mov     RAX,RDX
        jmp     .next
```

Now that we have all the numbers as ASCII characters we are nearly done. We now have to reverse the number in memory because the number saved at the lowest address is the digit with the least significance.

We now start with checking if we have written any character. If not then we write the ASCII character '0' into the memory. We use the instruction stosb for this to adjust the address in RDI at the same time.

44b ⟨*printdu64bit* 43a⟩+≡ (39a) ◁44a 44c▷

```
  .done:
        cmp     RDI,RSI
        jne     .printout
        mov     AL,'0'
        stosb
  .printout:
```

Next we calculate the number of digits that the number has. For this we move the address of the last digit to RDX and subtract the start of the scratch area from this. Next we adjust RDI because it points to the first address after the number.

44c ⟨*printdu64bit* 43a⟩+≡ (39a) ◁44b 44d▷

```
        mov     RDX,RDI
        sub     RDX,RSI
        dec     RDI
```

We now have RSI with the address of the start of the number and RDI with the address of the end. We now have to exchange the digits from the front and the end to get the right number. For this we increment RSI and decrement RDI after each exchange and when the addresses pass each other we are done.

44d ⟨*printdu64bit* 43a⟩+≡ (39a) ◁44c 45a▷

```
  .reverse:
        mov     AL,[RSI]
        mov     AH,[RDI]
        mov     [RSI],AH
        mov     [RDI],AL
        dec     RDI
        inc     RSI
        cmp     RSI,RDI
        jb      .reverse
```

Now we restore the address of the scratch area to RSI and move the number of digits (which we stored in RDX) to RDI and can the call _nprint to print the number.

45a ⟨*printdu64bit* 43a⟩+≡ (39a) ◁44d

```
    mov     RSI,R8
    mov     RDI,RDX
    call    _nprint
    ret
```

Uses _nprint 41b.

4.5.2 Printing a Hexadecimal 8bit Integer

The routine _printh8bit prints a given 8bit integer as hexadecimal number to stdout.

To print a hexadecimal number we mask a nibble (4bit) and have the number to print.

First we clear the register RAX and move the number to AX for further processing and clear the higher 8bit (AH). Additionally we move it to R8 for later restore.

Additionally we need the address of the scratch area in RDI for storing the result.

To increment the address during the processing we clear the direction flag.

Parameters

DI the number number to print to stdout. Only the lower 8bit are used.

RSI the address of a scratch area with a size of at least 2 bytes

45b ⟨*printh8bit* 45b⟩≡ (39a) 45c▷

```
_printh8bit:
    xor     RAX,RAX
    mov     AX,DI
    xor     AH,AH
    mov     R8,RAX
    mov     RDI,RSI
    cld
```

Defines:
_printh8bit, used in chunks 17c, 18a, 23b, 31b, and 32c.

Now we mask the higher 4 bit of AL by shifting it 4 bits to the right and mask out all but the lower 4 bit. Next we call the internal method printh8bit.printh4bit to print out this nibble.

45c ⟨*printh8bit* 45b⟩+≡ (39a) ◁45b 46a▷

```
    shr     AL,4
    and     AL,0x0f
    call    .printh4bit
```

Next we restore the number and print out the lower 4 bits.

46a ⟨*printh8bit* 45b⟩+≡ (39a) ◁45c

```
        mov     RAX,R8
        and     AL,0x0f
        call    .printh4bit
        mov     RDI,2
        call    _nprint
        ret
  ⟨printh8bit.printh4bit 46b⟩
```

Uses _nprint 41b.

Now we define the internal method to print a hexadecimal digit.

First we test if the digit is above or equal to 10. In this case we have to print out a character between `'a'` and `'f'` else we print out a decimal digit (between `'0'` and `'9'`).

Parameters (internal)

`AL`	the lower 4 bit contain the hexadecimal digit print to `stdout`
`RDI`	the address of a scratch area

46b ⟨*printh8bit.printh4bit* 46b⟩≡ (46a) 46c▷

```
  .printh4bit:
        cmp     AL,10
        jae     .printa2f
```

Defines:
`printh8bit.printh4bit`, never used.

Now we add `'0'` to get the code for the digit between `'0'` and `'9'`.

46c ⟨*printh8bit.printh4bit* 46b⟩+≡ (46a) ◁46b 46d▷

```
        add     AL,'0'
        jmp     .printout
```

Else we print a digit between `'a'` and `'f'`. We first subtract 10 because the value in `AL` is now between 10 and 15.

46d ⟨*printh8bit.printh4bit* 46b⟩+≡ (46a) ◁46c 46e▷

```
  .printa2f:
        sub     AL,10
        add     AL,'a'
```

Now we store the character into the storage area.

46e ⟨*printh8bit.printh4bit* 46b⟩+≡ (46a) ◁46d

```
  .printout:
        stosb
        ret
```

A Glossary

x86 x86 denotes a microprocessor architecture based on the 8086/8088 40

B Acronyms

ASCII American Standard Code for Information Interchange 43, 44

LF line feed 10, 11, 17

RNG random number generator 40

C x86-Instructions

`clflush` Flush Cache Line, introduced with Intel® Pentium® 4 10

`lfence` Load Fence, introduced with Intel® Pentium® 4 7, 10

`rdtsc` Read Time Stamp Counter, introduced with Intel® Pentium® 7, 9

D Code Chunks

⟨*analyzecachemintiming* 16a⟩
⟨*analyzecachesimpthrestiming* 20a⟩
⟨*cache-readback* 28c⟩
⟨*cacheread-program* 24⟩
⟨*cacheread-rodata* 30a⟩
⟨*cacheread-sample* 25c⟩
⟨*cacheread-udata* 25a⟩
⟨*cacheread.asm* 34b⟩
⟨*cachereadbyte-program* 13b⟩
⟨*cachereadbyte-rodata* 17a⟩
⟨*cachereadbyte.asm* 18c⟩
⟨*cachereadbyte2.asm* 21a⟩
⟨*cachereadbyte3-program* 21b⟩
⟨*cachereadbyte3.asm* 23c⟩
⟨*cachetiming-program* 9a⟩
⟨*cachetiming-rodata* 10c⟩
⟨*cachetiming.asm* 11f⟩
⟨*calculate-cache-access-time* 8a⟩
⟨*clear-cache* 26a⟩
⟨*clearcache* 13a⟩
⟨*common-rodata* 10d⟩
⟨*data-udata* 8b⟩
⟨*detect-byte* 27⟩
⟨*detect-cache-area-time* 26b⟩
⟨*enterstackframe* 39c⟩
⟨*exitProgram* 39b⟩
⟨*init-random-data* 9b⟩
⟨*init-random-probe* 13d⟩
⟨*leavestackframe* 39d⟩
⟨*license* 78⟩
⟨*nprint* 41b⟩
⟨*preamble* 5⟩
⟨*print* 42a⟩
⟨*print-comparision* 33b⟩
⟨*print-comparision16* 30b⟩
⟨*printdu64bit* 43a⟩
⟨*printh8bit* 45b⟩

⟨*printh8bit.printh4bit* 46b⟩
⟨*probe-udata* 13c⟩
⟨*readback-udata* 28b⟩
⟨*readbyte2cache* 22a⟩
⟨*readcachetiming* 14d⟩
⟨*scratch-udata* 8c⟩
⟨*timings-udata* 16b⟩
⟨*tsc-64bit* 7⟩
⟨*utilities* 39a⟩
⟨*xorshift-prng* 40a⟩

E License

E.1 GNU Free Documentation License

This license applies to this documentation as a whole.

```
                GNU Free Documentation License
                 Version 1.3, 3 November 2008


 Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
     <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other
functional and useful document "free" in the sense of freedom: to
assure everyone the effective freedom to copy and redistribute it,
with or without modifying it, either commercially or noncommercially.
Secondarily, this License preserves for the author and publisher a way
to get credit for their work, while not being considered responsible
for modifications made by others.

This License is a kind of "copyleft", which means that derivative
works of the document must themselves be free in the same sense.  It
complements the GNU General Public License, which is a copyleft
license designed for free software.

We have designed this License in order to use it for manuals for free
software, because free software needs free documentation: a free
program should come with manuals providing the same freedoms that the
software does.  But this License is not limited to software manuals;
it can be used for any textual work, regardless of subject matter or
whether it is published as a printed book.  We recommend this License
principally for works whose purpose is instruction or reference.


1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that
contains a notice placed by the copyright holder saying it can be
distributed under the terms of this License.  Such a notice grants a
```

world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain

ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

```
A. Use in the Title Page (and on the covers, if any) a title distinct
   from that of the Document, and from those of previous versions
   (which should, if there were any, be listed in the History section
   of the Document).  You may use the same title as a previous version
   if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities
   responsible for authorship of the modifications in the Modified
   Version, together with at least five of the principal authors of the
   Document (all of its principal authors, if it has fewer than five),
   unless they release you from this requirement.
C. State on the Title page the name of the publisher of the
   Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications
   adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice
   giving the public permission to use the Modified Version under the
   terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections
   and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add
   to it an item stating at least the title, year, new authors, and
   publisher of the Modified Version as given on the Title Page.  If
   there is no section Entitled "History" in the Document, create one
   stating the title, year, authors, and publisher of the Document as
   given on its Title Page, then add an item describing the Modified
   Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for
   public access to a Transparent copy of the Document, and likewise
   the network locations given in the Document for previous versions
   it was based on.  These may be placed in the "History" section.
   You may omit a network location for a work that was published at
   least four years before the Document itself, or if the original
   publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications",
   Preserve the Title of the section, and preserve in the section all
   the substance and tone of each of the contributor acknowledgements
   and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document,
   unaltered in their text and in their titles.  Section numbers
   or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements".  Such a section
   may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements"
   or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.
```

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements",

```
and any sections Entitled "Dedications".  You must delete all sections
Entitled "Endorsements".


6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other
documents released under this License, and replace the individual
copies of this License in the various documents with a single copy
that is included in the collection, provided that you follow the rules
of this License for verbatim copying of each of the documents in all
other respects.

You may extract a single document from such a collection, and
distribute it individually under this License, provided you insert a
copy of this License into the extracted document, and follow this
License in all other respects regarding verbatim copying of that
document.


7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate
and independent documents or works, in or on a volume of a storage or
distribution medium, is called an "aggregate" if the copyright
resulting from the compilation is not used to limit the legal rights
of the compilation's users beyond what the individual works permit.
When the Document is included in an aggregate, this License does not
apply to the other works in the aggregate which are not themselves
derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these
copies of the Document, then if the Document is less than one half of
the entire aggregate, the Document's Cover Texts may be placed on
covers that bracket the Document within the aggregate, or the
electronic equivalent of covers if the Document is in electronic form.
Otherwise they must appear on printed covers that bracket the whole
aggregate.


8. TRANSLATION

Translation is considered a kind of modification, so you may
distribute translations of the Document under the terms of section 4.
Replacing Invariant Sections with translations requires special
permission from their copyright holders, but you may include
translations of some or all Invariant Sections in addition to the
original versions of these Invariant Sections.  You may include a
translation of this License, and all the license notices in the
```

Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/licenses/.

Each version of the License is given a distinguishing version number.

If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
    Copyright (c)  YEAR  YOUR NAME.
    Permission is granted to copy, distribute and/or modify this document
```

```
    under the terms of the GNU Free Documentation License, Version 1.3
    or any later version published by the Free Software Foundation;
    with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
    A copy of the license is included in the section entitled "GNU
    Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts,
replace the "with...Texts." line with this:

    with the Invariant Sections being LIST THEIR TITLES, with the
    Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other
combination of the three, merge those two alternatives to suit the
situation.

If your document contains nontrivial examples of program code, we
recommend releasing these examples in parallel under your choice of
free software license, such as the GNU General Public License,
to permit their use in free software.
```

E.2 Code License

E.2.1 GNU GENERAL PUBLIC LICENSE

This license applies to all program code generated from this document.

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
```

price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section

```
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.
```

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

```
  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
```

```
The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

E.2.2 Code Chunk of GPL

This is a code chunk to be included by the generated asm files.

78 ⟨*license* 78⟩≡ (5)

```
;   Meltdown and Spectre - Samples Written in Assembly
;   Copyright (C) 2018 U. Plonus
;
;   This program is free software: you can redistribute it and/or modify
;   it under the terms of the GNU General Public License as published by
;   the Free Software Foundation, either version 3 of the License, or
;   (at your option) any later version.
;
;   This program is distributed in the hope that it will be useful,
;   but WITHOUT ANY WARRANTY; without even the implied warranty of
;   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;   GNU General Public License for more details.
;
;   You should have received a copy of the GNU General Public License
;   along with this program.  If not, see <http://www.gnu.org/licenses/>.
```