Zad 1.

Zadanie 1. Poniżej zamieszczono uproszczony wynik kompilacji pliku «data.c» do asemblera. Z jakimi opcjami[1] **sterownik kompilatora** (tj. polecenie gcc) wywołał kompilator języka C (tj. polecenie «cc1»), aby otrzymać plik «data.s»? Na podstawie dokumentu *GNU as: Assembler Directives*[2] wyjaśnij znaczenie **dyrektyw asemblera** użytych w poniższym kodzie. Które z dyrektyw przełączają bieżącą sekcję, dopisują zawartość do odpowiednich sekcji, modyfikują informacje przechowywane w nagłówku sekcji lub tablicę symboli?

```
 1     .globl foo            14     .globl bar            29     .globl some
 2     .data                 15     .bss                  30     .data
 3     .align 2              16     .align 8              31     .align 32
 4     .type foo, @object    17     .type bar, @object    32     .type some, @object
 5     .size foo, 2          18     .size bar, 8          33     .size some, 38
 6  foo:                     19  bar:                     34  some:
 7     .value 314            20     .zero 8               35     .quad weird
 8                           21                           36     .long -3
 9     .section .rodata      22     .globl array          37     .zero 4
10     .type abc, @object    23     .bss                  38     .quad abc
11     .size abc, 4          24     .align 32             39     .quad foo
12  abc:                     25     .type array, @object  40     .string "efghi"
13     .string "abc"         26     .size array, 800
                             27  array:
                             28     .zero 800
```

sterownik kompilatora — program wywołujący po kolei odpowiednie narzędzia potrzebne do przetworzenia programu (np. assembler, linker)

Compilation can involve up to four stages: preprocessing, compilation proper, assembly and linking,

-s

Stop after the stage of compilation proper; do not assemble. The output is in the form of an assembler code file for each non-assembler input file specified.

By default, the assembler file name for a source file is made by replacing the suffix '.c', '.i', etc., with '.s'.

Input files that don't require compilation are ignored.

```
/usr/lib/gcc/x86_64-linux-gnu/11/cc1 -quiet -v -imultiarch x86_64-linux-gnu data.c -quiet -dumpbase data.c -dumpb
ase-ext .c -mtune=generic -march=x86-64 -version -o data.s -fasynchronous-unwind-tables -fstack-protector-strong -
Wformat -Wformat-security -fstack-clash-protection -fcf-protection
```

gcc -v to pokazuje

Assembler directives are directions to the assembler to take some action or change a setting. Assembler directives do not represent instructions, and are not translated into machine code.
21 mar 2021

.global makes the symbol visible to ld. If you define *symbol* in your partial program, its value is made available to other partial programs that are linked with it. Otherwise, *symbol* takes its attributes from a symbol of the same name from another file linked into the same program.

.data tells as to assemble the following statements onto the end of the data subsection numbered *subsection* (which is an absolute expression). If *subsection* is omitted, it defaults to zero.

Pad the location counter (in the current subsection) to a particular storage boundary. The first expression (which must be absolute) is the alignment required, as described below. If this expression is omitted then a default value of 0 is used, effectively disabling alignment requirements.

Pad the location counter (in the current subsection) to a particular storage boundary. The first expression (which must be absolute) is the alignment required, as described below. If this expression is omitted then a default value of 0 is used, effectively disabling alignment requirements.

.align

```
short foo = 314;
```

7.102 .type

This directive is used to set the type of a symbol.

→

```
STT_OBJECT
object
```
Mark the symbol as being a data object.

ELF Version

For ELF targets, the .type directive is used like this:

.type *name* , *type description*

This sets the type of symbol *name* to be either a function symbol or an object symbol.

7.89 .size

This directive is used to set the size associated with a symbol.

znowu 2 bo short

7.85 .section *name*

Use the .section directive to assemble the following code into a section named *name*.

This directive is only supported for targets that actually support arbitrarily named sections; on a.out targets, for example, it is not accepted, even with a standard a.out section name.

7.94 .string "*str*", .string8 "*str*", .strir

"*str*", .string32 "*str*", .string64 "*str*"

Copy the characters in *str* to the object file.

7.112 .zero *size*

This directive emits *size* 0-valued bytes. *size* must be an absolute expression.

7.80 .quad *bignums*

.quad expects zero or more bignums, separated by commas. For each bignum, it emits an 8-byte integer.

analogicznie

7.62 .long *expressions*

.long is the same as '.int'. See .int.