Zadania to polega na napisaniu systemu operacyjnego który załaduję drugie etap systemu do pamięci a następnie go uruchomi. Drugi etap będzie zawierał macro którego ciało będzie znajdowało się w załączonej bibliotece którą, również należy napisać. Makro to ma np wypisywać znak na ekran. Zadania będzie składało się z następujących etapów:
a) Napisanie biblioteki zawierającej makro do wypisywania znaku na ekran.
b) Napisanie systemu który wywoła makro znajdującą się w załączonej bibliotece
c) Zmiana kodu tak aby wywołanie makra znajdowało się w niezaładowanej sekcji. (w celu obserwacji że nie działa bez odpowiedniego załadowania)
d) napisanie kodu który załaduję do pamięci drugą sekcję z wywołaniem makra.

Na początku tworzymy plik z rozszerzeniem .h będzie to nasza bibliotek.
Umieszczamy na początku dyrektywę
.altmacro

https://sourceware.org/binutils/docs/as/Altmacro.html

Następnie między dyrektywami .macro i .endm umieszczamy kody służacy do wypisywania znaku na ekran. Macro może przyjmować znak który ma wypisać lub zawsze wypisywać ten sam.

https://sourceware.org/binutils/docs/as/Macro.html

Teraz tworzymy plik .S, którego początek i koniec wyglądają analogicznie jak w poprzednim zadaniu choć tu możemy pominąć dyrektywy .org 510 i .word 0xaa55 Nimi zajmie się dołączony przeze mnie skrypt linkujący. W środku kodu znajdować ma się jedynie odwołanie do naszego macra aby sprawdzić czy wszystko działa.

Do kompilacji wykorzystamy gcc z opcjami:
-m32 //generuje kod dla odpowiedniego środowiska
-c //nie linkuje naszego pliku wyjściowego
-o //nazwa pliku wyjściowego z rozszerzeniem .o
na końcu nazwa pliku wejściowego

https://linux.die.net/man/1/gcc

linkowanie przy użyciu narzędzia ld
opcje:
-melf_i386
-nostdlib //przeszukuj tylko katalogi bibliotek wyraźnie określone w wierszu poleceń
-o //nazwa pliku wyjściowego z rozszerzeniem .elf
-T //nazwa skryptu linkującego ‘linker.ld’
plik wejściowy

Użyjemy teraz narzędzia objcopy
Kopiuje zawartość wejściowego pliku obiektowego do innego pliku, opcjonalnie zmieniając format pliku w tym procesie. Flaga
-O binary
pozwoli nam zmienić plik .elf na plik .img

https://linux.die.net/man/1/arm-linux-gnu-objcopy

Teraz wystarczy odpalić nasz obraz przez qemu i powinniśmy zobaczyć znak wypisany na ekranie.

Teraz umieścimy wywołanie makra w drugim etapie programu wpisując przed jego wywołaniem następujące dyrektywy.
.section .stage2
stage2:

w sekcji _start możemy umieścić instrukcję jmp stage2

Ponownie przechodzimy przez proces kompilacji linkowania i uruchomienia, widzimy że nie dostajemy takie samego wyniku jak poprzednio.
Musimy załadować stage do pamięci.

wykorzystamy w tym celu instrukcję
int $0x13

https://stanislavs.org/helppc/int_13-2.html

Przydatne informacje:
number of sectors 1
drive number 0x80
cylinder number 0
head number 0
sector number 2
pointer to buffer stage2

na końcu musimy wykonać jeszcze skok do nowo załadowanego etapu instrukcją jmp.

Jeśli wszystko zrobiliśmy poprawnie powinniśmy otrzymać efekt taki sam jak przy pierwszym uruchomieniu.