Az ARM előadás tematikája

- Miért éppen ARM?
- Áttekintés
  - processzor családok: ARM7, ARM9 és ARM11 vs. Cortex A5, A8, A9 és M0, M1, M3
  - „klasszikus” ARM vs. Cortex–M3
- Utasítások
  - ARM (32 bit), Thumb (16 bit), Thumb2 (16 vagy 32 bit)
  - váltás az utasításkészletek között
  - címek utolsó bitjei
- Regiszterkészlet
  - általános regiszterek
  - speciális regiszterek
  - státuszregiszterek
  - processzor regiszterek
- Processzor üzemmódok és vermek
  - Klasszikus ARM: supervisor, system, user, irq, fiq, undefined, abort
  - Cortex–M3: thread, hander, user
- Memóriatérkép
  - beépített FLASH, beépített RAM, perifériák és külső memóriák, rendszerperifériák
  - BOOT szekvencia, megszakítási rutinok
- Programfejlesztés kezdőlépései
  - `binutils`, `gcc`, `gdb`, és `libc` letöltése
  - `arm-none-linux-gnueabi-gcc` paraméterezése, a processzor kiválasztása
  - `Makefile` készítése
  - a megszakítási vektortábla elkészítése C nyelven, a verem beállítása
  - a linker script összeállítása a memóriatérkép alapján
- Letöltés
  - `OpenOCD` letöltése, fordítása, konfigurálása
  - bináris program FLASH-be írása
  - debug-olás `arm-none-linux-gnueabi-gdbtui`-val
- Programfejlesztés következő lépései
  - globális változók inicializálása, a `volatile` kulcsszó
  - perifériaregiszterek leképezés C struktúrákkal, pointerekkel
  - perifériakezelés
  - megszakításkezelés, megszakítási rutinok, top half, softirq, kölcsönös kizárás, szemaforok, üzenetsorok

C teszt: miért „3”-at ír ki?

```
#include <stdio.h>

int main(int argc, char **argv) {
    struct _t {
        int c, d;
        int *q[2];
    } a[2] = { {}, { .q = { &a[0].c, &a[0].d } } }, *b = a, **c = &b;

    *c[0][1].q[1] = 3;

    printf("%d\n", a[0].d);

    return 0;
}
```
