Zadanie 1

Program pozwalający znaleźć, które w miejsca w kodzie pochłaniają najwięcej cykli procesora oraz czasu. Także pokazuje ilość wykonań funkcji, drzewo wywołań, etc.

Flat profile :: Pokazuje ile czasu program spędza w danej funkcji oraz ile razy dana funkcja została wywołana.

Call graph :: Pokazuje drzewo wywołań, tj. która funkcja została wywołana i jaką funkcje wywołała ile razy itd.

Flaga pg dodaje dodatkowe informacje dla gprof'a. Linkuje także do binarki biblioteke libc_p

> The -pg flag causes gcc to do two additional things as it compiles your program. First, a small bit of code is added to the beginning of each function that records its address and the address it was called from at run time. gprof uses this data to generate a call graph. Second, gcc arranges to have a special "front end" added to the beginning of your program. The front end starts a recurring timer and every time the timer fires, 100 times per second, the currently executing address is saved. gprof uses this data to generate a "flat profile" showing where your program is spending its time.

Zadanie 2

```
13-6 VPN - Virtual Page Number
 5-0 VPO - Virtual Page Offset

13-8 TLBT (VPN) - TLB tag
 7-6 TLBI (VPN) - TLB index

11-6 PPN - Physical Page Number
 5-0 PPO - Physical Page Offset

11-6 CT (PPN) - Cache tag
 5-2 CI (PPO) - Cache index
 1-0 CO (PPO) - Byte offset within cache line

PPO = VPO

0x832 = 0b100000110010
0x835 = 0b100000110101
0xffd = 0b111111111101

Idx - 2 bits
Tag - 8 bits
Byte offset - 2 bits

1) Tag :: 10000011 :: 0x83
   Idx :: 00       :: 0x0
   Off :: 10       :: 0x2

   HIT! : Value 0xCC

2) Tag :: 10000011 :: 0x83
   Idx :: 01       :: 0x1
   Off :: 01       :: 0x1

   MISS
```

```
3) Tag :: 11111111 :: 0xff
   Idx :: 11       :: 3
   Off :: 01       :: 1

   HIT! : Value 0xC0
```

Zadanie 3

```
E -> B -> E -> D -> A -> E
A(0) B(1) C(2) D(3)

E(3) B(0) C(1) D(2)
E(2) B(3) C(0) D(1)
E(3) B(2) C(0) D(0)
E(2) B(1) C(0) D(3)
E(1) B(0) C(3) D(2)
E(3) B(0) C(2) D(1)
```

Zadanie 4

Block size - 4W 4x2row sections Memory - 1024W Address - 1W

Tag - 60 bits Off - 2 bits Idx - 2 bits

Najpierw 38 chybień zakładając, że nic wcześniej nie było załadowane do pamięci. Potem 8 chybień, bo przez resztę działania pętli dane będą załadowane.

$$Avg.time = ((38 + 8) * 100 + 8 * 4 * 5)/(38 + 5 * 8) = 61.02564ns$$

Zadanie 5

Write-back + write-allocate

Exclusive caches

L1 & L2

Write-back :: Dirty bit, defer write to mem until replacement of line. Information is only written to block in cache.

Dirty-bit :: Status bit which indicates whether the block is dirty (modified while in the cache) or clean (not modified). If clean then the block is not written ona miss.

Write-allocate :: Load into cache, update line in cache. Block is loaded on write miss.

Overall:

- On hits it writes cache setting "dirty" bit for the block, main memory is not updated.
- On misses it updates the block in main memory and brings the block to the cache
- Subsequant writes to the same block, if the block originally caused a miss, will hit in the cache next time, setting dirty bit for the block.