

Project: FreeRTOS: heap_4
Subject: Informatyka
Level: 1, Zakres Podstawowy, PP
Topic: dynamiczna alokacja pamięci, lista wolnych bloków, scalanie bloków
Revision date: 02.05.2026
Status: Draft
Card in series: 01/??
Card version: v0.1
Card key: mpabi/inf/rv32i-freertos/01/heap4/v0.1
UUID: 4654524c-73f2-46df-83a3-d448ba6d6c5a
Commit: e69aaa6d61af
Scope: pierwsza karta serii FreeRTOS; model dydaktyczny mechaniki heap_4.c
Files used: Task01: task01_heap_map.c
Task02: task02_free_list.c
Task03: task03_heap_init.c
Task04: task04_alignment.c
Task05: task05_malloc_first_fit.c
Task06: task06_split_block.c
Task07: task07_free_insert.c
Task08: task08_coalescing.c
Task09: task09_config_macros.c
Task10: task10_heap4_demo.c
Common: heap4_common.h
Runtime: memops.c
Startup: crt0.S

1 Cel

Ta karta zaczyna serię FreeRTOS od heap_4, ponieważ ten plik jest naturalnym pomostem po kartach z języka C. Wymaga wskaźników, tablic, struktur, listy jednokierunkowej, `static`, `typedef`, makr oraz arytmetyki adresów, ale nie wymaga jeszcze schedulera, przerwań ani uruchamiania tasków.

Przykłady są małym modelem edukacyjnym mechaniki heap_4.c. Nie są kopią pliku upstream FreeRTOS. Na RV32I wyniki są zapisywane w globalnych zmiennych `volatile`; `printf` jest dostępny wyłącznie w wariancie hostowym przez flagę `-DHOST_PRINTF=1`.

```
1 #ifndef HEAP4_COMMON_H
2 #define HEAP4_COMMON_H
3
4 #include <stddef.h>
5 #include <stdint.h>
6
7 #ifdef HOST_PRINTF
8 #include <stdio.h>
9 #define TRACE_PRINTF(...) printf(__VA_ARGS__)
10 #else
11 #define TRACE_PRINTF(...) ((void)0)
12 #endif
13
14 #define HEAP4_ARRAY_LEN(a) ((int)(sizeof(a) / sizeof((a)[0])))
```

```
15 #define HEAP4_ALIGNMENT ((size_t)8)
16 #define HEAP4_ALIGNMENT_MASK (HEAP4_ALIGNMENT - 1U)
17 #define HEAP4_ALLOCATED_BIT ((size_t)1U << (sizeof(size_t) * 8U - 1U))
18
19 typedef struct HeapBlock {
20     struct HeapBlock *next;
21     size_t size;
22 } HeapBlock;
23
24 static inline size_t heap4_align_up(size_t value)
25 {
26     if ((value & HEAP4_ALIGNMENT_MASK) != 0U)
27         value += HEAP4_ALIGNMENT - (value & HEAP4_ALIGNMENT_MASK);
28     return value;
29 }
30
31 static inline uintptr_t heap4_align_address(uintptr_t address)
32 {
33     uintptr_t mask;
34
35     mask = (uintptr_t)HEAP4_ALIGNMENT_MASK;
36     if ((address & mask) != 0U)
37         address += (uintptr_t)HEAP4_ALIGNMENT - (address & mask);
38     return address;
39 }
40
41 static inline size_t heap4_clear_allocated(size_t size)
42 {
43     return size & ~HEAP4_ALLOCATED_BIT;
44 }
45
46 static inline size_t heap4_mark_allocated(size_t size)
47 {
48     return size | HEAP4_ALLOCATED_BIT;
49 }
50
51 static inline int heap4_is_allocated(size_t size)
52 {
53     return (size & HEAP4_ALLOCATED_BIT) != 0U;
54 }
55
56 static inline int heap4_blocks_touch(HeapBlock *left, HeapBlock *right)
57 {
58     return (uint8_t *)left + heap4_clear_allocated(left->size) == (uint8_t *)right;
59 }
60
61 #endif
```

Listing 1: Wspólny nagłówek dla karty heap_4

1.1 static inline w nagłówku

Helpery z Listingu 1 są zapisane jako `static inline`, bo każdy task jest osobnym programem i łączy ten sam nagłówek. `static` daje prywatną kopię funkcji w każdym pliku `.c`, a `inline` pozwala kompilatorowi wstawić krótkie obliczenia, takie jak wyrównanie rozmiaru lub sprawdzenie bitu alokacji, bez kosztu zwykłego wywołania funkcji.

2 Mapa karty

Task	Temat
1	Mapa pojęć: obszar sterty, nagłówek bloku i bit alokacji.
2	Lista wolnych bloków jako lista jednokierunkowa.
3	Inicjalizacja sterty, pierwszy wolny blok i wartownik końca listy.
4	Wyrównanie rozmiarów żądań i adresu początku sterty.
5	Pierwszy pasujący blok, czyli zasada <code>first fit</code> .
6	Podział dużego bloku na część przydzieloną i pozostały blok wolny.
7	Zwracanie bloku i wstawianie go do listy według adresu.
8	Scalanie sąsiednich wolnych bloków.
9	Makra konfiguracyjne oraz hostowy <code>TRACE_PRINTF</code> .
10	Mini-demonstracja sekwencji <code>malloc/free</code> .

3 Task 01: mapa heap_4

Listing 2 pokazuje nagłówek bloku, wyrównanie rozmiaru i bit oznaczający blok zajęty.

```

1 #include "heap4_common.h"
2
3 volatile size_t g_header_size;
4 volatile size_t g_aligned_header_size;
5 volatile size_t g_allocated_bit_set;
6 volatile size_t g_user_bytes;
7 volatile size_t g_total_block_bytes;
8
9 int main(void)
10 {
11     HeapBlock block;
12     size_t requested;
13
14     requested = 13U;
15     block.next = 0;
16     block.size = heap4_mark_allocated(heap4_align_up(requested + sizeof(HeapBlock)));
17
18     g_header_size = sizeof(HeapBlock);
19     g_aligned_header_size = heap4_align_up(sizeof(HeapBlock));
20     g_allocated_bit_set = heap4_is_allocated(block.size);
21     g_total_block_bytes = heap4_clear_allocated(block.size);
22     g_user_bytes = g_total_block_bytes - sizeof(HeapBlock);
23
24     TRACE_PRINTF("header=%u aligned=%u allocated=%u total=%u user=%u\n",
25                 (unsigned)g_header_size, (unsigned)g_aligned_header_size,
26                 (unsigned)g_allocated_bit_set, (unsigned)g_total_block_bytes,
27                 (unsigned)g_user_bytes);

```

```

28     return 0;
29 }

```

Listing 2: Task 01: mapa heap_4

```

1  .file   "task01_heap_map.c"
2  .option nopic
3  .attribute arch, "rv32i2p1_zicsr2p0_zifencei2p0"
4  .attribute unaligned_access, 0
5  .attribute stack_align, 16
6  .text
7  .align  2
8  .globl  main
9  .type   main, @function
10 main:
11     li    a5,8
12     sw    a5,g_header_size,a4
13     sw    a5,g_aligned_header_size,a4
14     li    a5,1
15     sw    a5,g_allocated_bit_set,a4
16     lla   a5,g_total_block_bytes
17     li    a4,24
18     sw    a4,0(a5)
19     lw    a5,0(a5)
20     addi   a5,a5,-8
21     sw    a5,g_user_bytes,a4
22     li    a0,0
23     ret
24     .size  main, .-main
25     .globl g_total_block_bytes
26     .globl g_user_bytes
27     .globl g_allocated_bit_set
28     .globl g_aligned_header_size
29     .globl g_header_size
30     .section .sbss,"aw",@nobits
31     .align  2
32     .type   g_total_block_bytes, @object
33     .size   g_total_block_bytes, 4
34 g_total_block_bytes:
35     .zero   4
36     .type   g_user_bytes, @object
37     .size   g_user_bytes, 4
38 g_user_bytes:
39     .zero   4
40     .type   g_allocated_bit_set, @object
41     .size   g_allocated_bit_set, 4
42 g_allocated_bit_set:
43     .zero   4
44     .type   g_aligned_header_size, @object
45     .size   g_aligned_header_size, 4
46 g_aligned_header_size:
47     .zero   4
48     .type   g_header_size, @object
49     .size   g_header_size, 4
50 g_header_size:
51     .zero   4
52     .ident  "GCC: (15.2.0-23) 15.2.0"
53     .section .note.GNU-stack,"",@progbits

```

Listing 3: ASM dla task 01

4 Task 02: lista wolnych bloków

Listing 4 przechodzi po liście wolnych bloków i liczy sumę dostępnej pamięci.

```

1 #include "heap4_common.h"
2
3 volatile size_t g_free_total;
4 volatile size_t g_largest_free;

```

```
5 volatile int g_node_count;
6
7 static size_t list_total(HeapBlock *start)
8 {
9     HeapBlock *block;
10    size_t total;
11
12    total = 0U;
13    for (block = start->next; block != 0; block = block->next)
14        total += block->size;
15    return total;
16 }
17
18 static size_t list_largest(HeapBlock *start)
19 {
20    HeapBlock *block;
21    size_t largest;
22
23    largest = 0U;
24    for (block = start->next; block != 0; block = block->next)
25        if (block->size > largest)
26            largest = block->size;
27    return largest;
28 }
29
30 static int list_count(HeapBlock *start)
31 {
32    HeapBlock *block;
33    int count;
34
35    count = 0;
36    for (block = start->next; block != 0; block = block->next)
37        count++;
38    return count;
39 }
40
41 int main(void)
42 {
43    HeapBlock start;
44    HeapBlock a;
45    HeapBlock b;
46    HeapBlock c;
47
48    start.next = &a;
49    start.size = 0U;
50    a.next = &b;
51    a.size = 32U;
52    b.next = &c;
53    b.size = 80U;
54    c.next = 0;
55    c.size = 24U;
56
57    g_free_total = list_total(&start);
```

```

58     g_largest_free = list_largest(&start);
59     g_node_count = list_count(&start);
60
61     TRACE_PRINTF("free=%u largest=%u count=%d\n",
62                 (unsigned)g_free_total, (unsigned)g_largest_free, g_node_count);
63     return 0;
64 }

```

Listing 4: Task 02: lista wolnych bloków

```

1     .file    "task02_free_list.c"
2     .option nopic
3     .attribute arch, "rv32i2p1_zicsr2p0_zifencei2p0"
4     .attribute unaligned_access, 0
5     .attribute stack_align, 16
6     .text
7     .align 2
8     .globl main
9     .type main, @function
10 main:
11     addi    sp,sp,-32
12     addi    a5,sp,16
13     sw      a5,24(sp)
14     li      a5,32
15     sw      a5,28(sp)
16     addi    a5,sp,8
17     sw      a5,16(sp)
18     li      a5,80
19     sw      a5,20(sp)
20     sw      zero,8(sp)
21     li      a5,24
22     sw      a5,12(sp)
23     li      a4,0
24     add     a5,sp,a5
25 .L2:
26     lw      a3,4(a5)
27     add     a4,a4,a3
28     lw      a5,0(a5)
29     bne     a5,zero,.L2
30     sw      a4,g_free_total,a5
31     li      a3,0
32     addi    a5,sp,24
33     j       .L4
34 .L3:
35     lw      a5,0(a5)
36     beq     a5,zero,.L10
37 .L4:
38     lw      a4,4(a5)
39     bgeu    a3,a4,.L3
40     mv      a3,a4
41     j       .L3
42 .L10:
43     sw      a3,g_largest_free,a5
44     addi    a5,sp,24
45     li      a4,0
46 .L5:
47     addi    a4,a4,1
48     lw      a5,0(a5)
49     bne     a5,zero,.L5
50     sw      a4,g_node_count,a5
51     li      a0,0
52     addi    sp,sp,32
53     jr      ra
54     .size   main,.-main
55     .globl g_node_count
56     .globl g_largest_free
57     .globl g_free_total
58     .section .sbss,"aw",@nobits
59     .align 2
60     .type   g_node_count, @object
61     .size   g_node_count, 4
62 g_node_count:
63     .zero   4

```

```

64     .type    g_largest_free, @object
65     .size    g_largest_free, 4
66 g_largest_free:
67     .zero    4
68     .type    g_free_total, @object
69     .size    g_free_total, 4
70 g_free_total:
71     .zero    4
72     .ident   "GCC: (15.2.0-23) 15.2.0"
73     .section  .note.GNU-stack,"",@progbits

```

Listing 5: ASM dla task 02

5 Task 03: inicjalizacja sterty

Listing 6 tworzy pierwszy wolny blok oraz końcowy znacznik listy.

```

1  #include "heap4_common.h"
2
3  #define HEAP_BYTES 160U
4
5  static uint8_t heap_area[HEAP_BYTES + HEAP4_ALIGNMENT];
6  static HeapBlock start;
7  static HeapBlock *end_marker;
8
9  volatile size_t g_initial_free_size;
10 volatile uintptr_t g_aligned_offset;
11 volatile int g_end_is_last;
12
13 static void heap_init(void)
14 {
15     uintptr_t raw;
16     uintptr_t aligned;
17     size_t available;
18     HeapBlock *first;
19
20     raw = (uintptr_t)&heap_area[0];
21     aligned = heap4_align_address(raw);
22     available = HEAP_BYTES - (size_t)(aligned - raw);
23     available &= ~HEAP4_ALIGNMENT_MASK;
24
25     first = (HeapBlock *)aligned;
26     end_marker = (HeapBlock *)aligned + available / sizeof(HeapBlock);
27     end_marker->next = 0;
28     end_marker->size = 0U;
29
30     first->next = end_marker;
31     first->size = (uint8_t *)end_marker - (uint8_t *)first;
32
33     start.next = first;
34     start.size = 0U;
35 }
36
37 int main(void)
38 {
39     heap_init();

```

```

40
41     g_aligned_offset = (uintptr_t)start.next - (uintptr_t)&heap_area[0];
42     g_initial_free_size = start.next->size;
43     g_end_is_last = (start.next->next == end_marker) && (end_marker->next == 0);
44
45     TRACE_PRINTF("offset=%u free=%u end_last=%d\n",
46                 (unsigned)g_aligned_offset, (unsigned)g_initial_free_size,
47                 g_end_is_last);
48     return 0;
49 }

```

Listing 6: Task 03: inicjalizacja sterty

```

1     .file    "task03_heap_init.c"
2     .option nopic
3     .attribute arch, "rv32i2p1_zicsr2p0_zifencei2p0"
4     .attribute unaligned_access, 0
5     .attribute stack_align, 16
6     .text
7     .align   2
8     .globl   main
9     .type    main, @function
10  main:
11     lla      a2, .LANCHORO
12     andi     a4, a2, 7
13     mv       a5, a2
14     beq      a4, zero, .L2
15     andi     a5, a2, -8
16     addi     a5, a5, 8
17  .L2:
18     lla      a3, .LANCHORO+160
19     sub      a3, a3, a5
20     addi     a4, a5, -8
21     andi     a3, a3, -8
22     add      a4, a4, a3
23     sw       a4, end_marker, a3
24     sw       zero, 0(a4)
25     sw       zero, 4(a4)
26     sw       a4, 0(a5)
27     sub      a3, a4, a5
28     sw       a3, 4(a5)
29     lla      a1, start
30     sw       a5, 0(a1)
31     sw       zero, 4(a1)
32     sub      a5, a5, a2
33     sw       a5, g_aligned_offset, a2
34     sw       a3, g_initial_free_size, a5
35     lw       a5, 0(a4)
36     seqz     a5, a5
37     sw       a5, g_end_is_last, a4
38     li       a0, 0
39     ret
40     .size    main, .-main
41     .globl   g_end_is_last
42     .globl   g_aligned_offset
43     .globl   g_initial_free_size
44     .bss
45     .align   2
46     .set     .LANCHORO, . + 0
47     .type    heap_area, @object
48     .size    heap_area, 168
49  heap_area:
50     .zero    168
51     .section      .sbss, "aw", @nobits
52     .align   2
53     .type    g_end_is_last, @object
54     .size    g_end_is_last, 4
55  g_end_is_last:
56     .zero    4
57     .type    g_aligned_offset, @object
58     .size    g_aligned_offset, 4
59  g_aligned_offset:

```

```

60     .zero    4
61     .type    g_initial_free_size, @object
62     .size    g_initial_free_size, 4
63 g_initial_free_size:
64     .zero    4
65     .type    end_marker, @object
66     .size    end_marker, 4
67 end_marker:
68     .zero    4
69     .type    start, @object
70     .size    start, 8
71 start:
72     .zero    8
73     .ident   "GCC: (15.2.0-23) 15.2.0"
74     .section .note.GNU-stack,"",@progbits

```

Listing 7: ASM dla task 03

6 Task 04: wyrównanie

Listing 8 pokazuje, dlaczego alokator zaokrągla rozmiary i adres początku sterty.

```

1  #include "heap4_common.h"
2
3  volatile size_t g_request_1;
4  volatile size_t g_request_9;
5  volatile size_t g_request_17;
6  volatile uintptr_t g_address_delta;
7
8  int main(void)
9  {
10     uint8_t bytes[32];
11     uintptr_t raw;
12     uintptr_t aligned;
13
14     g_request_1 = heap4_align_up(1U + sizeof(HeapBlock));
15     g_request_9 = heap4_align_up(9U + sizeof(HeapBlock));
16     g_request_17 = heap4_align_up(17U + sizeof(HeapBlock));
17
18     raw = (uintptr_t)&bytes[1];
19     aligned = heap4_align_address(raw);
20     g_address_delta = aligned - raw;
21
22     TRACE_PRINTF("r1=%u r9=%u r17=%u delta=%u\n",
23                 (unsigned)g_request_1, (unsigned)g_request_9,
24                 (unsigned)g_request_17, (unsigned)g_address_delta);
25     return 0;
26 }

```

Listing 8: Task 04: wyrównanie

```

1  .file "task04_alignment.c"
2  .option nopic
3  .attribute arch, "rv32i2p1_zicsr2p0_zifencei2p0"
4  .attribute unaligned_access, 0
5  .attribute stack_align, 16
6  .text
7  .align 2
8  .globl main
9  .type main, @function
10 main:

```

```

11      addi    sp,sp,-32
12      li      a5,16
13      sw      a5,g_request_1,a4
14      li      a5,24
15      sw      a5,g_request_9,a4
16      li      a5,32
17      sw      a5,g_request_17,a4
18      li      a5,7
19      sw      a5,g_address_delta,a4
20      li      a0,0
21      addi    sp,sp,32
22      jr      ra
23      .size   main,.-main
24      .globl  g_address_delta
25      .globl  g_request_17
26      .globl  g_request_9
27      .globl  g_request_1
28      .section .sbss,"aw",@nobits
29      .align  2
30      .type   g_address_delta, @object
31      .size   g_address_delta, 4
32 g_address_delta:
33      .zero   4
34      .type   g_request_17, @object
35      .size   g_request_17, 4
36 g_request_17:
37      .zero   4
38      .type   g_request_9, @object
39      .size   g_request_9, 4
40 g_request_9:
41      .zero   4
42      .type   g_request_1, @object
43      .size   g_request_1, 4
44 g_request_1:
45      .zero   4
46      .ident  "GCC: (15.2.0-23) 15.2.0"
47      .section .note.GNU-stack,"",@progbits

```

Listing 9: ASM dla task 04

7 Task 05: first fit

Listing 10 wybiera pierwszy wolny blok wystarczająco duży dla żądania.

```

1  #include "heap4_common.h"
2
3  #define HEAP_BYTES 128U
4
5  static uint8_t heap_area[HEAP_BYTES + HEAP4_ALIGNMENT];
6  static HeapBlock start;
7  static HeapBlock *end_marker;
8
9  volatile size_t g_allocated_size;
10 volatile size_t g_remaining_free;
11 volatile int g_allocation_ok;
12
13 static void heap_init_one_block(void)
14 {
15     uintptr_t raw;
16     uintptr_t aligned;
17     HeapBlock *first;
18
19     raw = (uintptr_t)&heap_area[0];
20     aligned = heap4_align_address(raw);

```

```
21     first = (HeapBlock *)aligned;
22     end_marker = (HeapBlock *) (aligned + 120U);
23
24     end_marker->next = 0;
25     end_marker->size = 0U;
26     first->next = end_marker;
27     first->size = (uint8_t *)end_marker - (uint8_t *)first;
28     start.next = first;
29     start.size = 0U;
30 }
31
32 static void *heap_malloc_whole_block(size_t wanted)
33 {
34     HeapBlock *previous;
35     HeapBlock *block;
36     size_t total;
37
38     total = heap4_align_up(wanted + sizeof(HeapBlock));
39     previous = &start;
40     block = start.next;
41
42     while (block != end_marker) {
43         if (block->size >= total) {
44             previous->next = block->next;
45             block->next = 0;
46             block->size = heap4_mark_allocated(block->size);
47             return (uint8_t *)block + sizeof(HeapBlock);
48         }
49         previous = block;
50         block = block->next;
51     }
52
53     return 0;
54 }
55
56 int main(void)
57 {
58     void *p;
59     HeapBlock *allocated;
60
61     heap_init_one_block();
62     p = heap_malloc_whole_block(24U);
63     allocated = (HeapBlock *) ((uint8_t *)p - sizeof(HeapBlock));
64
65     g_allocation_ok = p != 0;
66     g_allocated_size = heap4_clear_allocated(allocated->size);
67     g_remaining_free = start.next->size;
68
69     TRACE_PRINTF("ok=%d allocated=%u remaining=%u\n",
70                 g_allocation_ok, (unsigned)g_allocated_size,
71                 (unsigned)g_remaining_free);
72     return 0;
73 }
```

Listing 10: Task 05: first fit

```

1      .file   "task05_malloc_first_fit.c"
2      .option nopic
3      .attribute arch, "rv32i2p1_zicsr2p0_zifencei2p0"
4      .attribute unaligned_access, 0
5      .attribute stack_align, 16
6      .text
7      .align  2
8      .globl  main
9      .type   main, @function
10     main:
11         lla    a4, .LANCHORO
12         andi   a5, a4, 7
13         beq    a5, zero, .L2
14         andi   a4, a4, -8
15         addi   a4, a4, 8
16     .L2:
17         mv     a5, a4
18         addi   a3, a4, 120
19         sw     a3, end_marker, a2
20         sw     zero, 120(a4)
21         sw     zero, 4(a3)
22         sw     a3, 0(a4)
23         li     a2, 120
24         sw     a2, 4(a4)
25         lla    a2, start
26         sw     a4, 0(a2)
27         sw     zero, 4(a2)
28         li     a1, 31
29         j      .L5
30     .L6:
31         mv     a5, a4
32     .L5:
33         lw     a4, 4(a5)
34         bgtu   a4, a1, .L7
35         lw     a4, 0(a5)
36         mv     a2, a5
37         bne    a4, a3, .L6
38         li     a5, 0
39         j      .L4
40     .L7:
41         lw     a4, 0(a5)
42         sw     a4, 0(a2)
43         sw     zero, 0(a5)
44         lw     a4, 4(a5)
45         li     a3, -2147483648
46         or     a4, a4, a3
47         sw     a4, 4(a5)
48         addi   a5, a5, 8
49     .L4:
50         snez   a4, a5
51         sw     a4, g_allocation_ok, a3
52         lw     a5, -4(a5)
53         slli   a5, a5, 1
54         srli   a5, a5, 1
55         sw     a5, g_allocated_size, a4
56         lw     a5, start
57         lw     a5, 4(a5)
58         sw     a5, g_remaining_free, a4
59         li     a0, 0
60         ret
61     .size     main, .-main
62     .globl    g_allocation_ok
63     .globl    g_remaining_free
64     .globl    g_allocated_size
65     .bss
66     .align  2
67     .set     .LANCHORO, . + 0
68     .type    heap_area, @object
69     .size    heap_area, 136
70     heap_area:
71         .zero 136
72     .section .sbss, "aw", @nobits

```

```

73     .align 2
74     .type g_allocation_ok, @object
75     .size g_allocation_ok, 4
76 g_allocation_ok:
77     .zero 4
78     .type g_remaining_free, @object
79     .size g_remaining_free, 4
80 g_remaining_free:
81     .zero 4
82     .type g_allocated_size, @object
83     .size g_allocated_size, 4
84 g_allocated_size:
85     .zero 4
86     .type end_marker, @object
87     .size end_marker, 4
88 end_marker:
89     .zero 4
90     .type start, @object
91     .size start, 8
92 start:
93     .zero 8
94     .ident "GCC: (15.2.0-23) 15.2.0"
95     .section .note.GNU-stack,"",@progbits

```

Listing 11: ASM dla task 05

8 Task 06: podział bloku

Listing 12 zostawia resztę dużego bloku na liście wolnych bloków.

```

1  #include "heap4_common.h"
2
3  #define HEAP_BYTES 192U
4  #define MIN_SPLIT_SIZE ((size_t)(sizeof(HeapBlock) * 2U))
5
6  static uint8_t heap_area[HEAP_BYTES + HEAP4_ALIGNMENT];
7  static HeapBlock start;
8  static HeapBlock *end_marker;
9
10 volatile size_t g_allocated_size;
11 volatile size_t g_split_free_size;
12 volatile int g_free_nodes;
13
14 static void heap_init_one_block(void)
15 {
16     uintptr_t aligned;
17     HeapBlock *first;
18
19     aligned = heap4_align_address((uintptr_t)&heap_area[0]);
20     first = (HeapBlock *)aligned;
21     end_marker = (HeapBlock *) (aligned + 176U);
22     end_marker->next = 0;
23     end_marker->size = 0U;
24     first->next = end_marker;
25     first->size = (uint8_t *)end_marker - (uint8_t *)first;
26     start.next = first;
27     start.size = 0U;
28 }
29
30 static void *heap_malloc_split(size_t wanted)

```

```
31 {
32     HeapBlock *previous;
33     HeapBlock *block;
34     HeapBlock *remainder;
35     size_t total;
36     size_t original;
37
38     total = heap4_align_up(wanted + sizeof(HeapBlock));
39     previous = &start;
40     block = start.next;
41
42     while (block != end_marker) {
43         if (block->size >= total) {
44             original = block->size;
45             if (original - total > MIN_SPLIT_SIZE) {
46                 remainder = (HeapBlock *)((uint8_t *)block + total);
47                 remainder->size = original - total;
48                 remainder->next = block->next;
49                 previous->next = remainder;
50                 block->size = total;
51             } else {
52                 previous->next = block->next;
53             }
54             block->next = 0;
55             block->size = heap4_mark_allocated(block->size);
56             return (uint8_t *)block + sizeof(HeapBlock);
57         }
58         previous = block;
59         block = block->next;
60     }
61
62     return 0;
63 }
64
65 static int count_free_nodes(void)
66 {
67     HeapBlock *block;
68     int count;
69
70     count = 0;
71     for (block = start.next; block != end_marker; block = block->next)
72         count++;
73     return count;
74 }
75
76 int main(void)
77 {
78     void *p;
79     HeapBlock *allocated;
80
81     heap_init_one_block();
82     p = heap_malloc_split(32U);
83     allocated = (HeapBlock *)((uint8_t *)p - sizeof(HeapBlock));
```

```

84
85     g_allocated_size = heap4_clear_allocated(allocated->size);
86     g_split_free_size = start.next->size;
87     g_free_nodes = count_free_nodes();
88
89     TRACE_PRINTF("allocated=%u split=%u nodes=%d\n",
90                 (unsigned)g_allocated_size, (unsigned)g_split_free_size,
91                 g_free_nodes);
92     return 0;
93 }

```

Listing 12: Task 06: podział bloku

```

1     .file    "task06_split_block.c"
2     .option nopie
3     .attribute arch, "rv32i2p1_zicsr2p0_zifencei2p0"
4     .attribute unaligned_access, 0
5     .attribute stack_align, 16
6     .text
7     .align 2
8     .globl main
9     .type main, @function
10 main:
11     lla     a4,.LANCHORO
12     andi    a5,a4,7
13     beq     a5,zero,.L2
14     andi    a4,a4,-8
15     addi    a4,a4,8
16 .L2:
17     mv      a5,a4
18     addi    a3,a4,176
19     sw      a3,end_marker,a2
20     sw      zero,176(a4)
21     sw      zero,4(a3)
22     sw      a3,0(a4)
23     li      a2,176
24     sw      a2,4(a4)
25     lla     a2,start
26     sw      a4,0(a2)
27     sw      zero,4(a2)
28     li      a1,39
29     j       .L7
30 .L10:
31     mv      a5,a4
32 .L7:
33     lw      a4,4(a5)
34     bgtu    a4,a1,.L13
35     lw      a4,0(a5)
36     mv      a2,a5
37     bne     a4,a3,.L10
38     li      a5,0
39     j       .L6
40 .L13:
41     addi    a4,a4,-40
42     li      a1,16
43     bleu    a4,a1,.L4
44     sw      a4,44(a5)
45     lw      a4,0(a5)
46     sw      a4,40(a5)
47     addi    a4,a5,40
48     sw      a4,0(a2)
49     li      a4,40
50     sw      a4,4(a5)
51 .L5:
52     sw      zero,0(a5)
53     lw      a4,4(a5)
54     li      a2,-2147483648
55     or      a4,a4,a2
56     sw      a4,4(a5)
57     addi    a5,a5,8
58 .L6:
59     lw      a5,-4(a5)

```

```

60      slli    a5,a5,1
61      srli    a5,a5,1
62      sw      a5,g_allocated_size,a4
63      lw      a5,start
64      lw      a4,4(a5)
65      sw      a4,g_split_free_size,a2
66      beq     a5,a3,.L11
67      li      a4,0
68  .L9:
69      addi    a4,a4,1
70      lw      a5,0(a5)
71      bne     a5,a3,.L9
72  .L8:
73      sw      a4,g_free_nodes,a5
74      li      a0,0
75      ret
76  .L4:
77      lw      a4,0(a5)
78      sw      a4,0(a2)
79      j       .L5
80  .L11:
81      li      a4,0
82      j       .L8
83      .size   main, .-main
84      .globl  g_free_nodes
85      .globl  g_split_free_size
86      .globl  g_allocated_size
87      .bss
88      .align  2
89      .set    .LANCHORO, . + 0
90      .type   heap_area, @object
91      .size   heap_area, 200
92  heap_area:
93      .zero   200
94      .section .sbss,"aw",@nobits
95      .align  2
96      .type   g_free_nodes, @object
97      .size   g_free_nodes, 4
98  g_free_nodes:
99      .zero   4
100     .type   g_split_free_size, @object
101     .size   g_split_free_size, 4
102  g_split_free_size:
103     .zero   4
104     .type   g_allocated_size, @object
105     .size   g_allocated_size, 4
106  g_allocated_size:
107     .zero   4
108     .type   end_marker, @object
109     .size   end_marker, 4
110  end_marker:
111     .zero   4
112     .type   start, @object
113     .size   start, 8
114  start:
115     .zero   8
116     .ident   "GCC: (15.2.0-23) 15.2.0"
117     .section .note.GNU-stack,"",@progbits

```

Listing 13: ASM dla task 06

9 Task 07: free i wstawianie

Listing 14 zwraca blok do listy tak, aby lista pozostała posortowana po adresach.

```

1  #include "heap4_common.h"
2
3  static uint8_t heap_area[192];
4  static HeapBlock start;
5  static HeapBlock *end_marker;

```

```
6
7 volatile int g_order_ok;
8 volatile int g_free_count;
9 volatile size_t g_total_free;
10
11 static void insert_block(HeapBlock *block)
12 {
13     HeapBlock *iterator;
14
15     for (iterator = &start; iterator->next < block; iterator = iterator->next)
16         ;
17
18     block->next = iterator->next;
19     iterator->next = block;
20 }
21
22 static void heap_free_model(void *p)
23 {
24     HeapBlock *block;
25
26     if (p == 0)
27         return;
28
29     block = (HeapBlock *)((uint8_t *)p - sizeof(HeapBlock));
30     block->size = heap4_clear_allocated(block->size);
31     insert_block(block);
32 }
33
34 static int count_free(void)
35 {
36     HeapBlock *block;
37     int count;
38
39     count = 0;
40     for (block = start.next; block != end_marker; block = block->next)
41         count++;
42     return count;
43 }
44
45 static size_t total_free(void)
46 {
47     HeapBlock *block;
48     size_t total;
49
50     total = 0U;
51     for (block = start.next; block != end_marker; block = block->next)
52         total += block->size;
53     return total;
54 }
55
56 int main(void)
57 {
58     HeapBlock *a;
```

```

59     HeapBlock *b;
60     HeapBlock *c;
61     void *payload;
62
63     a = (HeapBlock *)&heap_area[0];
64     b = (HeapBlock *)&heap_area[64];
65     c = (HeapBlock *)&heap_area[128];
66     end_marker = (HeapBlock *)&heap_area[176];
67
68     start.next = a;
69     a->next = c;
70     c->next = end_marker;
71     end_marker->next = 0;
72
73     a->size = 40U;
74     b->size = heap4_mark_allocated(32U);
75     c->size = 48U;
76     end_marker->size = 0U;
77
78     payload = (uint8_t *)b + sizeof(HeapBlock);
79     heap_free_model(payload);
80
81     g_order_ok = start.next == a && a->next == b && b->next == c && c->next ==
end_marker;
82     g_free_count = count_free();
83     g_total_free = total_free();
84
85     TRACE_PRINTF("order=%d count=%d total=%u\n",
86                 g_order_ok, g_free_count, (unsigned)g_total_free);
87     return 0;
88 }

```

Listing 14: Task 07: free i wstawianie

```

1     .file    "task07_free_insert.c"
2     .option nopic
3     .attribute arch, "rv32i2p1_zicsr2p0_zifencei2p0"
4     .attribute unaligned_access, 0
5     .attribute stack_align, 16
6     .text
7     .align 2
8     .globl main
9     .type    main, @function
10 main:
11     lla     a5, .LANCHOR0
12     lla     a4, .LANCHOR0+176
13     sw      a4, end_marker, a3
14     sw      a5, start, a3
15     lla     a3, .LANCHOR0+128
16     sw      a3, 0(a5)
17     sw      a4, 128(a5)
18     sw      zero, 176(a5)
19     li      a4, 40
20     sw      a4, 4(a5)
21     li      a4, 48
22     sw      a4, 132(a5)
23     sw      zero, 180(a5)
24     li      a4, 32
25     sw      a4, 68(a5)
26     lla     a5, start
27     lla     a3, .LANCHOR0+64
28 .L2:
29     mv      a4, a5

```

```

30      lw      a5,0(a5)
31      bltu    a5,a3,.L2
32      lla     a3,.LANCHOR0
33      sw      a5,64(a3)
34      lla     a5,.LANCHOR0+64
35      sw      a5,0(a4)
36      lw      a5,start
37      beq     a5,a3,.L16
38      sw      zero,g_order_ok,a4
39      lla     a4,.LANCHOR0+176
40      beq     a5,a4,.L5
41  .L9:
42      mv      a4,a5
43      li      a3,0
44      lla     a2,.LANCHOR0+176
45  .L6:
46      addi    a3,a3,1
47      lw      a4,0(a4)
48      bne     a4,a2,.L6
49      sw      a3,g_free_count,a4
50      li      a4,0
51      lla     a2,.LANCHOR0+176
52  .L7:
53      lw      a3,4(a5)
54      add     a4,a4,a3
55      lw      a5,0(a5)
56      bne     a5,a2,.L7
57  .L8:
58      sw      a4,g_total_free,a5
59      li      a0,0
60      ret
61  .L16:
62      lla     a3,.LANCHOR0+64
63      lw      a2,.LANCHOR0
64      li      a4,0
65      beq     a2,a3,.L17
66  .L4:
67      sw      a4,g_order_ok,a3
68      j       .L9
69  .L17:
70      lla     a3,.LANCHOR0+128
71      lw      a2,.LANCHOR0+64
72      bne     a2,a3,.L4
73      lla     a3,.LANCHOR0+176
74      lw      a4,.LANCHOR0+128
75      sub     a4,a4,a3
76      seqz    a4,a4
77      j       .L4
78  .L5:
79      sw      zero,g_free_count,a5
80      li      a4,0
81      j       .L8
82      .size   main,.-main
83      .globl  g_total_free
84      .globl  g_free_count
85      .globl  g_order_ok
86      .bss
87      .align  2
88      .set    .LANCHOR0,. + 0
89      .type   heap_area, @object
90      .size   heap_area, 192
91  heap_area:
92      .zero   192
93      .section .sbss,"aw",@nobits
94      .align  2
95      .type   g_total_free, @object
96      .size   g_total_free, 4
97  g_total_free:
98      .zero   4
99      .type   g_free_count, @object
100     .size   g_free_count, 4
101  g_free_count:
102     .zero   4
103     .type   g_order_ok, @object
104     .size   g_order_ok, 4
105  g_order_ok:

```

```
106     .zero    4
107     .type    end_marker, @object
108     .size    end_marker, 4
109 end_marker:
110     .zero    4
111     .type    start, @object
112     .size    start, 8
113 start:
114     .zero    8
115     .ident   "GCC: (15.2.0-23) 15.2.0"
116     .section .note.GNU-stack,"",@progbits
```

Listing 15: ASM dla task 07

10 Task 08: scalanie bloków

Listing 16 scala blok z lewym i prawym sąsiadem, gdy adresy są ciągłe.

```
1  #include "heap4_common.h"
2
3  static uint8_t heap_area[192];
4  static HeapBlock start;
5  static HeapBlock *end_marker;
6
7  volatile int g_free_count;
8  volatile size_t g_merged_size;
9  volatile int g_merged_with_right;
10
11 static void insert_block_merge(HeapBlock *block)
12 {
13     HeapBlock *iterator;
14     uint8_t *block_end;
15
16     for (iterator = &start; iterator->next < block; iterator = iterator->next)
17         ;
18
19     block_end = (uint8_t *)block + block->size;
20     if (block_end == (uint8_t *)iterator->next) {
21         block->size += iterator->next->size;
22         block->next = iterator->next->next;
23     } else {
24         block->next = iterator->next;
25     }
26
27     if (iterator != &start && heap4_blocks_touch(iterator, block)) {
28         iterator->size += block->size;
29         iterator->next = block->next;
30     } else {
31         iterator->next = block;
32     }
33 }
34
35 static int count_free(void)
36 {
37     HeapBlock *block;
38     int count;
```

```

39     count = 0;
40     for (block = start.next; block != end_marker; block = block->next)
41         count++;
42     return count;
43 }
44
45
46 int main(void)
47 {
48     HeapBlock *left;
49     HeapBlock *middle;
50     HeapBlock *right;
51
52     left = (HeapBlock *)&heap_area[0];
53     middle = (HeapBlock *)&heap_area[48];
54     right = (HeapBlock *)&heap_area[96];
55     end_marker = (HeapBlock *)&heap_area[160];
56
57     start.next = left;
58     left->next = right;
59     right->next = end_marker;
60     end_marker->next = 0;
61
62     left->size = 48U;
63     middle->size = 48U;
64     right->size = 64U;
65     end_marker->size = 0U;
66
67     insert_block_merge(middle);
68
69     g_free_count = count_free();
70     g_merged_size = start.next->size;
71     g_merged_with_right = start.next->next == end_marker;
72
73     TRACE_PRINTF("count=%d size=%u merged_right=%d\n",
74                 g_free_count, (unsigned)g_merged_size, g_merged_with_right);
75     return 0;
76 }

```

Listing 16: Task 08: scalanie bloków

```

1     .file    "task08_coalescing.c"
2     .option nopic
3     .attribute arch, "rv32i2p1_zicsr2p0_zifencei2p0"
4     .attribute unaligned_access, 0
5     .attribute stack_align, 16
6     .text
7     .align 2
8     .globl main
9     .type    main, @function
10 main:
11     lla     a5, .LANCHOR0
12     lla     a4, .LANCHOR0+160
13     sw      a4, end_marker, a3
14     sw      a5, start, a3
15     lla     a3, .LANCHOR0+96
16     sw      a3, 0(a5)
17     sw      a4, 96(a5)
18     sw      zero, 160(a5)
19     li      a4, 48

```

```

20      sw      a4,4(a5)
21      sw      a4,52(a5)
22      li      a4,64
23      sw      a4,100(a5)
24      sw      zero,164(a5)
25      lla     a5,start
26      lla     a3,.LANCHOR0+48
27.L2:
28      mv      a4,a5
29      lw      a5,0(a5)
30      bltu    a5,a3,.L2
31      lla     a3,.LANCHOR0+96
32      beq     a5,a3,.L12
33.L3:
34      sw      a5,.LANCHOR0+48,a3
35      lla     a5,start
36      lla     a3,.LANCHOR0+48
37      beq     a4,a5,.L4
38      lw      a2,4(a4)
39      slli    a5,a2,1
40      srli    a5,a5,1
41      add     a5,a4,a5
42      beq     a5,a3,.L13
43.L4:
44      sw      a3,0(a4)
45      lw      a2,start
46      lla     a5,.LANCHOR0+160
47      beq     a2,a5,.L9
48      mv      a5,a2
49      li      a4,0
50      lla     a3,.LANCHOR0+160
51.L6:
52      addi    a4,a4,1
53      lw      a5,0(a5)
54      bne     a5,a3,.L6
55.L5:
56      sw      a4,g_free_count,a5
57      lw      a5,4(a2)
58      sw      a5,g_merged_size,a4
59      lw      a5,0(a2)
60      lla     a4,.LANCHOR0+160
61      sub     a5,a5,a4
62      seqz    a5,a5
63      sw      a5,g_merged_with_right,a4
64      li      a0,0
65      ret
66.L12:
67      li      a5,112
68      sw      a5,.LANCHOR0+52,a3
69      lw      a5,0(a4)
70      lw      a5,0(a5)
71      j       .L3
72.L13:
73      lla     a3,.LANCHOR0
74      lw      a5,52(a3)
75      add     a5,a5,a2
76      sw      a5,4(a4)
77      lw      a3,48(a3)
78      j       .L4
79.L9:
80      li      a4,0
81      j       .L5
82      .size   main,.-main
83      .globl  g_merged_with_right
84      .globl  g_merged_size
85      .globl  g_free_count
86      .bss
87      .align  2
88      .set    .LANCHOR0,. + 0
89      .type   heap_area, @object
90      .size   heap_area, 192
91 heap_area:
92      .zero   192
93      .section .sbss,"aw",@nobits
94      .align  2
95      .type   g_merged_with_right, @object

```

```

96     .size    g_merged_with_right, 4
97 g_merged_with_right:
98     .zero    4
99     .type    g_merged_size, @object
100    .size    g_merged_size, 4
101 g_merged_size:
102    .zero    4
103    .type    g_free_count, @object
104    .size    g_free_count, 4
105 g_free_count:
106    .zero    4
107    .type    end_marker, @object
108    .size    end_marker, 4
109 end_marker:
110    .zero    4
111    .type    start, @object
112    .size    start, 8
113 start:
114    .zero    8
115    .ident    "GCC: (15.2.0-23) 15.2.0"
116    .section    .note.GNU-stack,"",@progbits

```

Listing 17: ASM dla task 08

11 Task 09: makra konfiguracyjne

Listing 18 pokazuje małe odpowiedniki makr konfiguracyjnych FreeRTOS-a.

```

1  #include "heap4_common.h"
2
3  #ifndef CONFIG_TOTAL_HEAP_SIZE
4  #define CONFIG_TOTAL_HEAP_SIZE 256U
5  #endif
6
7  #ifndef CONFIG_SUPPORT_DYNAMIC_ALLOCATION
8  #define CONFIG_SUPPORT_DYNAMIC_ALLOCATION 1
9  #endif
10
11 #define CONFIG_ADJUSTED_HEAP_SIZE (CONFIG_TOTAL_HEAP_SIZE - HEAP4_ALIGNMENT)
12
13 volatile size_t g_total_heap;
14 volatile size_t g_adjusted_heap;
15 volatile int g_dynamic_enabled;
16 volatile int g_trace_is_host_only;
17
18 int main(void)
19 {
20     g_total_heap = CONFIG_TOTAL_HEAP_SIZE;
21     g_adjusted_heap = CONFIG_ADJUSTED_HEAP_SIZE;
22     g_dynamic_enabled = CONFIG_SUPPORT_DYNAMIC_ALLOCATION;
23
24     #ifdef HOST_PRINTF
25         g_trace_is_host_only = 1;
26     #else
27         g_trace_is_host_only = 0;
28     #endif
29
30     TRACE_PRINTF("heap=%u adjusted=%u dynamic=%d host_trace=%d\n",
31                 (unsigned)g_total_heap, (unsigned)g_adjusted_heap,

```

```

32         g_dynamic_enabled, g_trace_is_host_only);
33     return 0;
34 }

```

Listing 18: Task 09: makra konfiguracyjne

```

1     .file    "task09_config_macros.c"
2     .option nopic
3     .attribute arch, "rv32i2p1_zicsr2p0_zifencei2p0"
4     .attribute unaligned_access, 0
5     .attribute stack_align, 16
6     .text
7     .align 2
8     .globl main
9     .type    main, @function
10    main:
11        li    a5,256
12        sw    a5,g_total_heap,a4
13        li    a5,248
14        sw    a5,g_adjusted_heap,a4
15        li    a5,1
16        sw    a5,g_dynamic_enabled,a4
17        sw    zero,g_trace_is_host_only,a5
18        li    a0,0
19        ret
20        .size  main, .-main
21        .globl g_trace_is_host_only
22        .globl g_dynamic_enabled
23        .globl g_adjusted_heap
24        .globl g_total_heap
25        .section    .sbss,"aw",@nobits
26        .align 2
27        .type  g_trace_is_host_only, @object
28        .size  g_trace_is_host_only, 4
29    g_trace_is_host_only:
30        .zero 4
31        .type  g_dynamic_enabled, @object
32        .size  g_dynamic_enabled, 4
33    g_dynamic_enabled:
34        .zero 4
35        .type  g_adjusted_heap, @object
36        .size  g_adjusted_heap, 4
37    g_adjusted_heap:
38        .zero 4
39        .type  g_total_heap, @object
40        .size  g_total_heap, 4
41    g_total_heap:
42        .zero 4
43        .ident "GCC: (15.2.0-23) 15.2.0"
44        .section    .note.GNU-stack,"",@progbits

```

Listing 19: ASM dla task 09

12 Task 10: mini heap_4

Listing 20 łączy inicjalizację, alokację, zwalnianie i scalanie w jeden model.

```

1    #include "heap4_common.h"
2
3    #define HEAP_BYTES 256U
4    #define MIN_SPLIT_SIZE ((size_t)(sizeof(HeapBlock) * 2U))
5
6    static uint8_t heap_area[HEAP_BYTES + HEAP4_ALIGNMENT];
7    static HeapBlock start;
8    static HeapBlock *end_marker;
9
10   volatile int g_allocations_ok;

```

```
11 volatile int g_final_free_nodes;
12 volatile size_t g_final_free_total;
13 volatile size_t g_minimum_ever_free;
14
15 static size_t free_bytes_remaining;
16 static size_t minimum_ever_free;
17
18 static size_t list_total(void)
19 {
20     HeapBlock *block;
21     size_t total;
22
23     total = 0U;
24     for (block = start.next; block != end_marker; block = block->next)
25         total += block->size;
26     return total;
27 }
28
29 static int list_count(void)
30 {
31     HeapBlock *block;
32     int count;
33
34     count = 0;
35     for (block = start.next; block != end_marker; block = block->next)
36         count++;
37     return count;
38 }
39
40 static void insert_block_merge(HeapBlock *block)
41 {
42     HeapBlock *iterator;
43     uint8_t *block_end;
44
45     for (iterator = &start; iterator->next < block; iterator = iterator->next)
46         ;
47
48     block_end = (uint8_t *)block + block->size;
49     if (block_end == (uint8_t *)iterator->next) {
50         block->size += iterator->next->size;
51         block->next = iterator->next->next;
52     } else {
53         block->next = iterator->next;
54     }
55
56     if (iterator != &start && heap4_blocks_touch(iterator, block)) {
57         iterator->size += block->size;
58         iterator->next = block->next;
59     } else {
60         iterator->next = block;
61     }
62 }
63
```

```
64 static void heap_init(void)
65 {
66     uintptr_t raw;
67     uintptr_t aligned;
68     size_t available;
69     HeapBlock *first;
70
71     raw = (uintptr_t)&heap_area[0];
72     aligned = heap4_align_address(raw);
73     available = HEAP_BYTES - (size_t)(aligned - raw);
74     available &= ~HEAP4_ALIGNMENT_MASK;
75
76     first = (HeapBlock *)aligned;
77     end_marker = (HeapBlock *)(aligned + available - sizeof(HeapBlock));
78     end_marker->next = 0;
79     end_marker->size = 0U;
80
81     first->next = end_marker;
82     first->size = (uint8_t *)end_marker - (uint8_t *)first;
83     start.next = first;
84     start.size = 0U;
85
86     free_bytes_remaining = first->size;
87     minimum_ever_free = free_bytes_remaining;
88 }
89
90 static void *heap_malloc_model(size_t wanted)
91 {
92     HeapBlock *previous;
93     HeapBlock *block;
94     HeapBlock *remainder;
95     size_t total;
96     size_t original;
97
98     if (wanted == 0U)
99         return 0;
100
101     total = heap4_align_up(wanted + sizeof(HeapBlock));
102     previous = &start;
103     block = start.next;
104
105     while (block != end_marker) {
106         if (block->size >= total) {
107             original = block->size;
108             if (original - total > MIN_SPLIT_SIZE) {
109                 remainder = (HeapBlock *)((uint8_t *)block + total);
110                 remainder->size = original - total;
111                 remainder->next = block->next;
112                 previous->next = remainder;
113                 block->size = total;
114             } else {
115                 previous->next = block->next;
116                 total = original;
```

```
117     }
118
119     block->next = 0;
120     block->size = heap4_mark_allocated(block->size);
121     free_bytes_remaining -= total;
122     if (free_bytes_remaining < minimum_ever_free)
123         minimum_ever_free = free_bytes_remaining;
124     return (uint8_t *)block + sizeof(HeapBlock);
125 }
126 previous = block;
127 block = block->next;
128 }
129
130 return 0;
131 }
132
133 static void heap_free_model(void *p)
134 {
135     HeapBlock *block;
136     size_t size;
137
138     if (p == 0)
139         return;
140
141     block = (HeapBlock *)((uint8_t *)p - sizeof(HeapBlock));
142     if (!heap4_is_allocated(block->size))
143         return;
144
145     size = heap4_clear_allocated(block->size);
146     block->size = size;
147     free_bytes_remaining += size;
148     insert_block_merge(block);
149 }
150
151 int main(void)
152 {
153     void *a;
154     void *b;
155     void *c;
156
157     heap_init();
158
159     a = heap_malloc_model(24U);
160     b = heap_malloc_model(40U);
161     heap_free_model(a);
162     c = heap_malloc_model(16U);
163     heap_free_model(b);
164     heap_free_model(c);
165
166     g_allocations_ok = a != 0 && b != 0 && c != 0;
167     g_final_free_nodes = list_count();
168     g_final_free_total = list_total();
169     g_minimum_ever_free = minimum_ever_free;
```

```

170
171     TRACE_PRINTF("ok=%d nodes=%d free=%u min=%u\n",
172                 g_allocations_ok, g_final_free_nodes,
173                 (unsigned)g_final_free_total, (unsigned)g_minimum_ever_free);
174     return 0;
175 }

```

Listing 20: Task 10: mini heap_4

```

1     .file    "task10_heap4_demo.c"
2     .option nopic
3     .attribute arch, "rv32i2p1_zicsr2p0_zifencei2p0"
4     .attribute unaligned_access, 0
5     .attribute stack_align, 16
6     .text
7     .align 2
8     .type    heap_malloc_model, @function
9 heap_malloc_model:
10     beq      a0,zero,.L1
11     addi     a4,a0,8
12     andi     a0,a0,7
13     beq      a0,zero,.L3
14     andi     a4,a4,-8
15     addi     a4,a4,8
16 .L3:
17     lw       a0,start
18     lw       a3,end_marker
19     beq      a0,a3,.L10
20     lla      a2,start
21     j        .L8
22 .L11:
23     mv       a0,a5
24 .L8:
25     lw       a5,4(a0)
26     bgeu     a5,a4,.L12
27     lw       a5,0(a0)
28     mv       a2,a0
29     bne      a5,a3,.L11
30     li       a0,0
31     ret
32 .L12:
33     sub      a3,a5,a4
34     li       a1,16
35     bleu     a3,a1,.L5
36     add      a5,a0,a4
37     sw       a3,4(a5)
38     lw       a3,0(a0)
39     sw       a3,0(a5)
40     sw       a5,0(a2)
41     sw       a4,4(a0)
42 .L6:
43     sw       zero,0(a0)
44     lw       a5,4(a0)
45     li       a3,-2147483648
46     or       a5,a5,a3
47     sw       a5,4(a0)
48     lla      a3,free_bytes_remaining
49     lw       a5,0(a3)
50     sub      a5,a5,a4
51     sw       a5,0(a3)
52     lw       a4,minimum_ever_free
53     bgeu     a5,a4,.L7
54     sw       a5,minimum_ever_free,a4
55 .L7:
56     addi     a0,a0,8
57     ret
58 .L5:
59     lw       a4,0(a0)
60     sw       a4,0(a2)
61     mv       a4,a5
62     j        .L6
63 .L10:
64     li       a0,0
65 .L1:

```

```

66         ret
67         .size    heap_malloc_model, .-heap_malloc_model
68         .align   2
69         .type    heap_free_model, @function
70 heap_free_model:
71         beq      a0,zero,.L13
72         lw       a5,-4(a0)
73         blt      a5,zero,.L19
74 .L13:
75         ret
76 .L19:
77         addi     a3,a0,-8
78         slli     a5,a5,1
79         srli     a2,a5,1
80         sw       a2,-4(a0)
81         lla      a4,free_bytes_remaining
82         lw       a5,0(a4)
83         add      a5,a5,a2
84         sw       a5,0(a4)
85         lla      a5,start
86 .L15:
87         mv       a4,a5
88         lw       a5,0(a5)
89         bgtu     a3,a5,.L15
90         add      a1,a3,a2
91         beq      a5,a1,.L20
92 .L16:
93         sw       a5,-8(a0)
94         lla      a5,start
95         beq      a4,a5,.L17
96         lw       a2,4(a4)
97         slli     a5,a2,1
98         srli     a5,a5,1
99         add      a5,a4,a5
100        beq      a3,a5,.L21
101 .L17:
102        sw       a3,0(a4)
103        ret
104 .L20:
105        lw       a5,4(a5)
106        add      a5,a5,a2
107        sw       a5,-4(a0)
108        lw       a5,0(a4)
109        lw       a5,0(a5)
110        j        .L16
111 .L21:
112        lw       a5,-4(a0)
113        add      a5,a5,a2
114        sw       a5,4(a4)
115        lw       a5,-8(a0)
116        sw       a5,0(a4)
117        ret
118        .size    heap_free_model, .-heap_free_model
119        .align   2
120        .globl   main
121        .type    main, @function
122 main:
123        addi     sp,sp,-32
124        sw       ra,28(sp)
125        sw       s0,24(sp)
126        sw       s1,20(sp)
127        sw       s2,16(sp)
128        sw       s3,12(sp)
129        sw       s4,8(sp)
130        lla      a4,.LANCHOR0
131        andi     a3,a4,7
132        mv       a5,a4
133        beq      a3,zero,.L23
134        andi     a4,a4,-8
135        addi     a5,a4,8
136 .L23:
137        lla      s0,.LANCHOR0+256
138        sub      s0,s0,a5
139        andi     s0,s0,-8
140        addi     a4,a5,-8
141        add      s0,s0,a4

```

```

142     sw     s0,end_marker,a4
143     sw     zero,0(s0)
144     sw     zero,4(s0)
145     sw     s0,0(a5)
146     sub    a4,s0,a5
147     sw     a4,4(a5)
148     lla    s4,start
149     sw     a5,0(s4)
150     sw     zero,4(s4)
151     sw     a4,free_bytes_remaining,a5
152     sw     a4,minimum_ever_free,a5
153     li     a0,24
154     call   heap_malloc_model
155     mv     s2,a0
156     li     a0,40
157     call   heap_malloc_model
158     mv     s3,a0
159     mv     a0,s2
160     call   heap_free_model
161     li     a0,16
162     call   heap_malloc_model
163     mv     s1,a0
164     mv     a0,s3
165     call   heap_free_model
166     mv     a0,s1
167     call   heap_free_model
168     snez   s2,s2
169     snez   s3,s3
170     and    s2,s2,s3
171     snez   s1,s1
172     and    s1,s1,s2
173     sw     s1,g_allocations_ok,a5
174     lw     a5,0(s4)
175     beq    s0,a5,.L24
176     mv     a4,a5
177     li     a3,0
178 .L25:
179     addi   a3,a3,1
180     lw     a4,0(a4)
181     bne    s0,a4,.L25
182     sw     a3,g_final_free_nodes,a2
183     li     a3,0
184 .L26:
185     lw     a2,4(a5)
186     add    a3,a3,a2
187     lw     a5,0(a5)
188     bne    a5,a4,.L26
189 .L27:
190     sw     a3,g_final_free_total,a5
191     lw     a5,minimum_ever_free
192     sw     a5,g_minimum_ever_free,a4
193     li     a0,0
194     lw     ra,28(sp)
195     lw     s0,24(sp)
196     lw     s1,20(sp)
197     lw     s2,16(sp)
198     lw     s3,12(sp)
199     lw     s4,8(sp)
200     addi   sp,sp,32
201     jr     ra
202 .L24:
203     sw     zero,g_final_free_nodes,a5
204     li     a3,0
205     j      .L27
206     .size  main,.-main
207     .globl g_minimum_ever_free
208     .globl g_final_free_total
209     .globl g_final_free_nodes
210     .globl g_allocations_ok
211     .bss
212     .align 2
213     .set   .LANCHORO,. + 0
214     .type  heap_area, @object
215     .size  heap_area, 264
216 heap_area:
217     .zero  264

```

```
218     .section          .sbss,"aw",@nobits
219     .align   2
220     .type    minimum_ever_free, @object
221     .size    minimum_ever_free, 4
222 minimum_ever_free:
223     .zero    4
224     .type    free_bytes_remaining, @object
225     .size    free_bytes_remaining, 4
226 free_bytes_remaining:
227     .zero    4
228     .type    g_minimum_ever_free, @object
229     .size    g_minimum_ever_free, 4
230 g_minimum_ever_free:
231     .zero    4
232     .type    g_final_free_total, @object
233     .size    g_final_free_total, 4
234 g_final_free_total:
235     .zero    4
236     .type    g_final_free_nodes, @object
237     .size    g_final_free_nodes, 4
238 g_final_free_nodes:
239     .zero    4
240     .type    g_allocations_ok, @object
241     .size    g_allocations_ok, 4
242 g_allocations_ok:
243     .zero    4
244     .type    end_marker, @object
245     .size    end_marker, 4
246 end_marker:
247     .zero    4
248     .type    start, @object
249     .size    start, 8
250 start:
251     .zero    8
252     .ident   "GCC: (15.2.0-23) 15.2.0"
253     .section          .note.GNU-stack,"",@progbits
```

Listing 21: ASM dla task 10

13 Polecenia

1. Zbuduj kartę dla RV32I poleceniem `make tasks`.
2. Zbuduj wariant hostowy poleceniem `make host`; wypisywanie działa tylko przez `HOST_PRINTF`.
3. W tasku 6 zmień próg `MIN_SPLIT_SIZE` i sprawdź, kiedy blok nie jest dzielony.
4. W tasku 8 zmień rozmiary bloków tak, aby scalanie działało tylko z prawym sąsiadem.
5. W tasku 10 dodaj większą alokację, która ma się nie udać, i zapisz status w nowej zmiennej `volatile`.