	TITLE 4: From Linear Cursor to Reusable Blocks	VER. v00.01	DATETIME 2026-07-20T00:00:00+02:00		
	PROJECT Freestanding C nad FreeRTOS	SERIES FreeRTOS C	CARD 01/15	SHEET 1/7	
	SUBJ. Inf. PROG. _ CORE _ SCOPE _ LEVEL _ POS. _	GITEA_UUID CARD_UUID aea09ab7-9bfd-5368-8d1b-c8724871c1a7	87e48095-7793-54e3-9230-4e61d31bd459 AUTHOR M. Paliszczak		
	GITEA https://zsl-gitea.mpabi.pl/edu-freertos-c/lab-rv32i-freertos-heap4				
	HTML https://dce7fb9d-7b2f-5d49-96a2-3a30d3070b84.mpabi.pl/aea09ab7-9bfd-5368-8d1b-c8724871c1a7				

Cel karty

Uczeń przechodzi od liniowego kursora z K&R 5.4 do wielokrotnego użycia tej samej areny: rozpoznaje nagłówek bloku, first-fit, split oraz dwustronne scalanie wolnych bloków w heap_4.

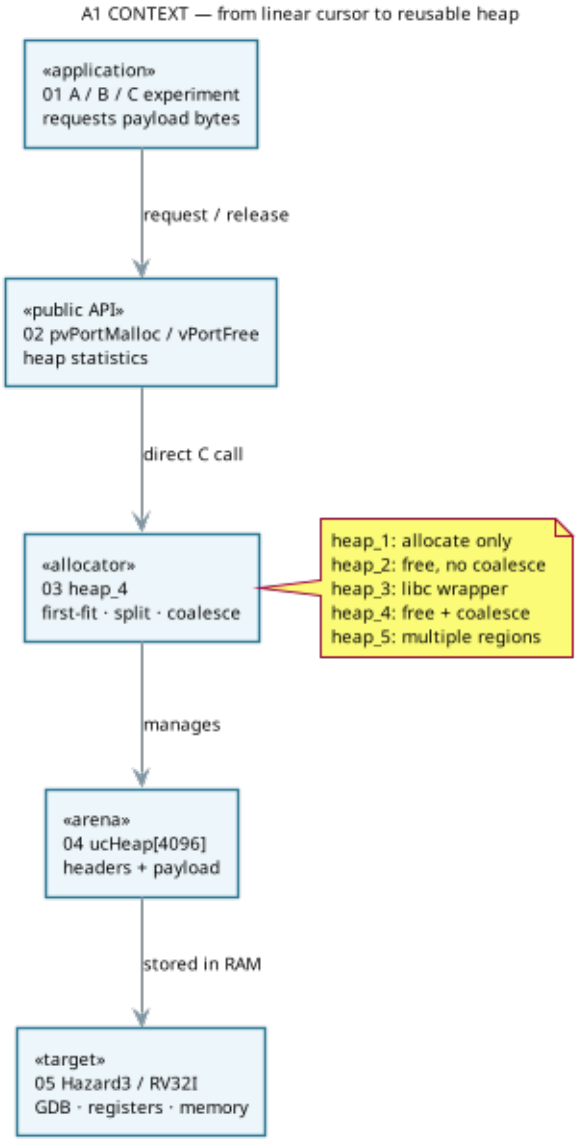
Zakres karty

Dwa małe modele C przygotowują przewidywanie. Główny replay uruchamia prawdziwe FreeRTOS-Kernel V11.3.0 heap_4 na RV32I/Hazard3 i dowodzi fragmentacji, pełnego scalenia, minimum-ever oraz kontrolowanego OOM.

Lekcja	Task	Najważniejsza idea	Priorytet	Status	Version
FC01	Task01	**headers, address-ordered free list, first-fit, split, coalescing and heap statistics**	**główny**	**opracowane**	v00.01

A1 — Od liniowego kursora do odzyskiwalnej areny

część 1/1 · r1 c1



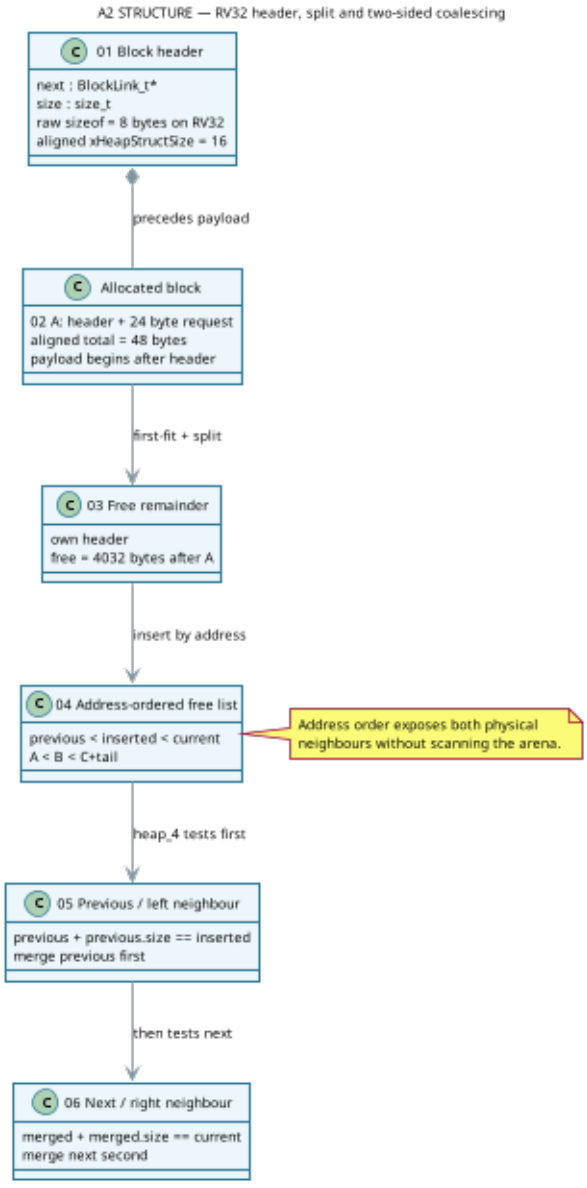
Rysunek 1: A1 CONTEXT — od żądania aplikacji do rzeczywistych bajtów ucHeap. — część 1/1 (r1 c1).

INTERPRETACJA

Aplikacja korzysta z publicznego API, heap_4 zarządza jedną areną, a Hazard3 pokazuje jej rzeczywisty stan.

A2 — Nagłówek, payload i lista wolnych bloków

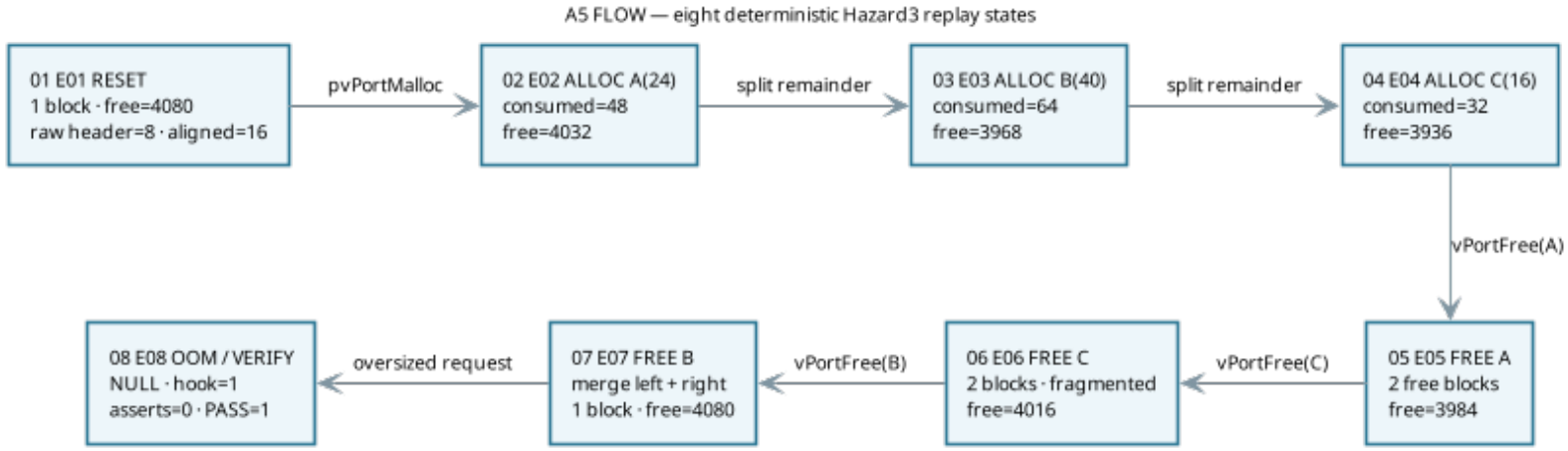
część 1/1 · r1 c1



Rysunek 2: A2 STRUCTURE — metadane wewnątrz areny i adresowo uporządkowana lista. — część 1/1 (r1 c1).

INTERPRETACJA
Każdy blok ma nagłówek, a tylko wolne bloki wykorzystują wskaźnik next do budowy listy uporządkowanej adresami.

A5 — First-fit, fragmentacja i pełne scalenie



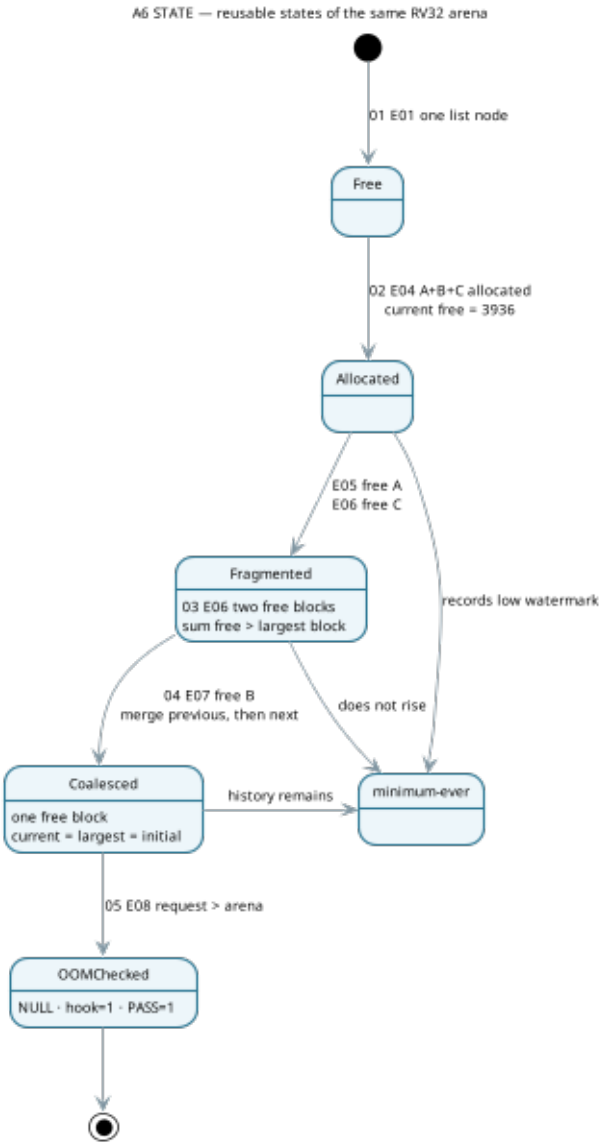
Rysunek 3: A5 FLOW — inicjalizacja, A/B/C, fragmentacja, dwustronne scalenie i OOM. — część 1/1 (r1 c1).

INTERPRETACJA

Jeden przebieg A/B/C przechodzi od pustej areny przez trzy przydziały i dwa wolne obszary do ponownie scalonego bloku.

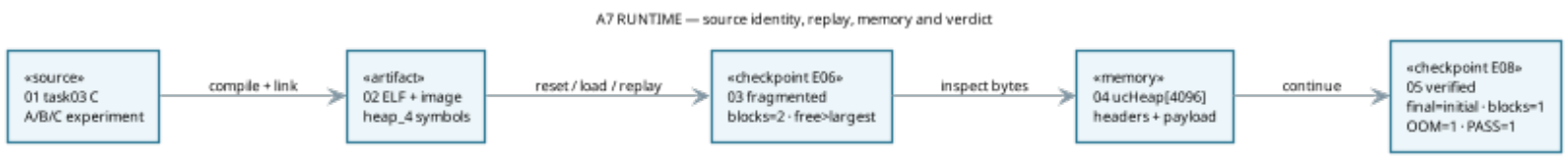
A6 — Stan bloku a stan statystyk

część 1/1 · r1 c1



Rysunek 4: A6 STATE — wolny, przydzielony, rozdzielony, scalony oraz minimum-ever. — część 1/1 (r1 c1).

INTERPRETACJA
Bieżąca liczba wolnych bajtów może rosnąć po free, ale minimum-ever pozostaje historycznym minimum.



Rysunek 5: A7 RUNTIME — kod, ELF, checkpoint, pamięć i statystyki jednego przebiegu. — część 1/1 (r1 c1).

INTERPRETACJA

Ten sam artefakt łączy źródło, symbole ELF, osiem checkpointów, pamięć ucHeap i końcowe asercje.

1 Task01 — mechanika heap_4

TASK TASK01 · Predict and prove reusable heap_4 blocks

aea09ab7-9bfd-5368-8d1b-c8724871c1a7

BLOCK A - przewidywanie

Przejdź przez model nagłówka, split i porządek adresowy przed uruchomieniem prawdziwego heap_4.

STEP size · Policz rozmiary całych bloków.

Rozróżnij surowy BlockLink_t (8 bajtów) od xHeapStructSize (16 bajtów), dodaj nagłówek do żądań 24, 40 i 16 bajtów, a następnie wyrównaj wynik do 16.

STEP list · Narysuj listę po free(A,C).

Zaznacz, dlaczego przydzielony B rozdziela dwa wolne obszary.

BLOCK B - replay Hazard3

Powiąż każdy stan diagramu z deterministycznym checkpointem E01–E08.

STEP allocate · Sprawdź E01-E04.

Potwierdź wartości 4080, 4032, 3968 i 3936 oraz zużycie 48/64/32 bajtów.

STEP release · Sprawdź E05-E07.

Porównaj liczbę wolnych bloków i udowodnij scalenie poprzedniego, a potem następnego sąsiada.

STEP verify · Sprawdź E08.

Udowodnij zachowanie minimum-ever, kontrolowany OOM, brak asercji i PASS=1.

BLOCK Ćwiczenie · Ćwiczenie - ta sama suma, inna użyteczność

Zaproponuj dwa układy wolnych bloków o tej samej sumie bajtów, ale tylko jeden zdolny obsłużyć wskazane duże żądanie. Uzasadnij odpowiedź przez largest-free-block.

Evidence: Dwa diagramy listy, suma wolnych bajtów, rozmiar największego bloku i wynik pvPortMalloc.

Acceptance: Uczeń nie utożsamia sumy wolnej pamięci z możliwością pojedynczego dużego przydziału.

Task acceptance: Na RV32 BlockLink_t ma 8 bajtów, lecz 16-bajtowe wyrównanie portu daje xHeapStructSize=16; A/B/C zużywają 48/64/32 bajty; po free(A,C) są dwa wolne bloki, po free(B) jeden; final free = initial free; minimum-ever nie rośnie; OOM zwraca NULL; PASS=1.

WNIOSEK

heap_4 odzyskuje pamięć dzięki metadanym w arenie, adresowo uporządkowanej liście i scalaniu sąsiadów. Suma wolnych bajtów nie zastępuje informacji o największym bloku, a minimum-ever jest historią, nie bieżącym stanem.