

# Niskopoziomowa emulacja sprzętowa implementacja architektury systemowej konsoli Game Boy

Patryk Michalak

18 lutego 2026

## **Streszczenie**

Your abstract.

# Spis treści

<b>1 Wstęp</b>	<b>3</b>
1.1 Opis problemu . . . . .	3
1.2 Emulatory . . . . .	3
1.3 Założenia i cel pracy . . . . .	3
1.4 Grupa docelowa . . . . .	4
<b>2 Analiza architektury systemowej konsoli Game Boy</b>	<b>4</b>
2.1 Specyfikacja techniczna . . . . .	4
2.2 Procesor i SoC . . . . .	4
2.2.1 Model czasowy systemu . . . . .	4
2.3 Zestaw Instrukcji . . . . .	5
2.3.1 Instrukcje transferu danych . . . . .	5
2.3.2 Instrukcje arytmetyczne . . . . .	5
2.3.3 Instrukcje logiczne i bitowe . . . . .	7
2.3.4 Instrukcje przesunięć i rotacji . . . . .	7
2.3.5 Instrukcje sterowania przepływem programu . . . . .	7
2.4 Mapowanie pamięci w konsoli Game Boy . . . . .	7
2.4.1 Ogólny układ przestrzeni adresowej . . . . .	8
2.4.2 Pamięć programu i bankowanie ROM . . . . .	8
2.4.3 Pamięć wideo . . . . .	8
2.4.4 Pamięć robocza i obszar echo . . . . .	8
2.4.5 Mapowanie urządzeń wejścia/wyjścia . . . . .	8
2.4.6 Pamięć wysokiej prędkości . . . . .	8
2.4.7 Znaczenie architektury mapowania pamięci . . . . .	8
2.5 System przerwań w konsoli Game Boy . . . . .	9
2.5.1 Źródła przerwań . . . . .	9
2.5.2 Rejestry sterujące systemem przerwań . . . . .	9
2.5.3 Mechanizm obsługi przerwania . . . . .	9
2.5.4 Priorytety przerwań . . . . .	9
2.5.5 Rola systemu przerwań w architekturze konsoli . . . . .	10
2.6 Płyta główna . . . . .	10
2.6.1 Jednostka centralna . . . . .	10
2.6.2 Pamięć operacyjna i pamięć wideo . . . . .	10
2.6.3 Układ graficzny . . . . .	11
2.6.4 Układ dźwiękowy . . . . .	11
2.6.5 Interfejs kartridżu . . . . .	11
2.6.6 Układy wejścia i wyjścia . . . . .	11
2.7 Kartridże systemu Game Boy . . . . .	11
2.7.1 Kartridże bez kontrolera banków pamięci (ROM Only) . . . . .	12
2.7.2 Kartridże z kontrolerem MBC1 . . . . .	12
2.7.3 Kartridże z kontrolerem MBC2 . . . . .	12
2.7.4 Kartridże z kontrolerem MBC3 . . . . .	12
2.7.5 Kartridże z kontrolerem MBC5 . . . . .	12
2.7.6 Kartridże specjalizowane . . . . .	12
2.7.7 Struktura fizyczna kartridżu . . . . .	12

# 1 Wstęp

## 1.1 Opis problemu

Konsola *Game Boy*, wyprodukowana przez firmę *Nintendo* i wprowadzona na rynek japoński 21 kwietnia 1989 roku, stała się jednym z najbardziej rozpoznawalnych urządzeń w historii elektronicznej rozrywki. *Game Boy* odniósła znaczący sukces komercyjny – w ciągu pierwszych lat od premiery sprzedano ponad 20 milionów egzemplarzy. W kolejnych latach konsola została wprowadzona na rynki zachodnie, gdzie również zdobyła dużą popularność. Łączna sprzedaż wszystkich wariantów urządzenia uczyniła ją jedną z najlepiej sprzedających się konsol przenośnych w historii.

Gry dla konsoli były dystrybuowane w postaci wymiennych kartridżów (Game Pak), zawierających pamięć ROM z kodem programu, danymi graficznymi oraz dźwiękowymi. W niektórych przypadkach kartridż wyposażone były dodatkowo w pamięć RAM oraz baterię podtrzymującą stan gry. Architektura sprzętowa konsoli opierała się na 8-bitowym procesorze taktowanym częstotliwością około 4,19 MHz, a obraz generowany był w rozdzielcości  $160 \times 144$  pikseli przy czterech poziomach szarości.

Wraz z zakończeniem produkcji konsoli przez *Nintendo* dostęp do oryginalnego sprzętu stał się ograniczony. Znaczna część tytułów wydanych wyłącznie na tę platformę nie została oficjalnie przeniesiona na nowsze systemy. W konsekwencji wiele gier pozostaje dostępnych jedynie poprzez zachowane egzemplarze konsoli oraz kartridże, co utrudnia ich użytkowanie oraz archiwizację.

## 1.2 Emulatory

Jednym z rozwiązań problemu ograniczonej dostępności oryginalnego sprzętu są emulatory, czyli programy komputerowe umożliwiające odtworzenie funkcjonalności danego systemu sprzętowego w środowisku wirtualnym. Emulator odwzorowuje działanie procesora, pamięci, układów graficznych oraz urządzeń wejścia/wyjścia w sposób pozwalający na uruchamianie oryginalnego oprogramowania.

Emulatory znajdują zastosowanie nie tylko w środowisku graczy, lecz również w badaniach nad architekturą systemów komputerowych, analizie kompatybilności oprogramowania oraz cyfrowej archiwizacji dziedzictwa technologicznego. Umożliwiają one zachowanie historycznej wartości gier i systemów, które w przeciwnym razie mogłyby zostać utracone wraz z degradacją fizycznych nośników.

Wyróżnia się dwa podstawowe podejścia do emulacji:

- **Emulacja pełna** – polegająca na wiernym odwzorowaniu wszystkich kluczowych komponentów sprzętowych systemu, w tym procesora, pamięci, grafiki oraz dźwięku.
- **Emulacja częściowa** – obejmująca jedynie wybrane elementy systemu, co może ograniczać kompatybilność z oryginalnym oprogramowaniem.

W kontekście niniejszej pracy przyjęto założenie realizacji emulacji pełnej w zakresie podstawowej funkcjonalności konsoli.

## 1.3 Założenia i cel pracy

Celem niniejszej pracy jest zaprojektowanie oraz implementacja emulatora konsoli *Game Boy* w języku programowania C++, z wykorzystaniem biblioteki Raylib do obsługi warstwy graficznej i wejścia użytkownika. Projekt zakłada odwzorowanie podstawowych mechanizmów działania oryginalnej architektury sprzętowej w stopniu umożliwiającym uruchamianie wybranej klasy gier.

Zakres pracy obejmuje:

- implementację emulacji procesora oraz mapy pamięci konsoli,
- odwzorowanie mechanizmu wyświetlania obrazu,
- obsługę wejścia użytkownika z wykorzystaniem klawiatury,
- możliwość wczytywania obrazów ROM typu *ROM Only*,
- zapewnienie wieloplatformowości (Windows, Linux, macOS).

W ramach projektu przyjęto ograniczenie do obsługi podstawowego typu kartридża zawierającego wyłącznie pamięć ROM, bez dodatkowych kontrolerów pamięci (MBC). Emulator przeznaczony jest do uruchamiania cyfrowych kopii gier pozyskanych przez użytkownika z posiadanych nośników.

Poprawność działania systemu zostanie zweryfikowana poprzez porównanie wyników emulacji z publicznie dostępnymi testami zgodności opracowanymi przez społeczność dokumentującą architekturę konsoli.

## 1.4 Grupa docelowa

Odbiorcami opracowanego systemu są użytkownicy posiadający legalne kopie gier przeznaczonych na konsolę Game Boy, zainteresowani ich uruchamianiem na współczesnych komputerach osobistych. Emulator może stanowić również narzędzie edukacyjne dla osób zainteresowanych architekturą systemów wbudowanych oraz mechanizmami niskopoziomowego działania sprzętu.

# 2 Analiza architektury systemowej konsoli Game Boy

## 2.1 Specyfikacja techniczna

Nintendo Research And Development 1, pod przewodnictwem Gunpei Yokoi i Satoru Okady, zaprojektowało 8-bitową konsolę Game Boya. Charakteryzuje się wyświetlaczem matrycowym o rozdzielcości 160x144 pikseli, D-padem, czterema przyciskami gry i jednym głośnikiem. Używa stworzonych na potrzebe konsoli kartridżów ‘GamePak’. Zasilany był przez 4 baterie AA. Gracze mogli korzystać z kabla Game Link Cable do połączenia dwóch Game Boyów dla rozgrywki wieloosobowej bądź transferu danych dla wspierających gier. Chociaż wyświetlacz monochromatyczny był gorszy niż u konkurencji, umożliwił on bardziej tanie i długotrwałe życie baterii. [2]

## 2.2 Procesor i SoC

Procesorem Gameboya był specjalnie wytworzony na potrzeby konsoli 8-bitowy procesor SM83 firmy Sharp. [4]. Procesor był mieszaną dwóch innych procesorów - 8080 firmy Intel i Z80 firmy Zilog. Procesor znajdował się na płytce wraz innymi komponentami takimi jak pamięć RAM i ROM. Cała płytki jest określana jako SoC (System on Chip) i oryginalny Game Boy zawierał DMG-CPU znany także jako Sharp LR35902 [1]. SoC był wpinany do płyty głównej, która zawierała dodatkową pamięć RAM i Video RAM (VRAM) jak i łączyła się z innymi komponentami jak ekran, głośnik czy kontroler.

### 2.2.1 Model czasowy systemu

Działanie konsoli Game Boy oparte jest na synchronicznym modelu czasowym, w którym wszystkie główne komponenty systemu — procesor, układ graficzny oraz timery sprzętowe — pracują w oparciu o wspólne źródło taktowania. Nominalna częstotliwość zegara systemowego wynosi 4,194304 MHz, co determinuje tempo wykonywania instrukcji procesora oraz przebieg cykli pracy pozostałych bloków funkcjonalnych.

**Cykle maszynowe procesora** Procesor wykonuje instrukcje w jednostkach zwanych cyklami maszynowymi (ang. *machine cycles*), z których każdy obejmuje określoną liczbę taktów zegara systemowego. Czas wykonania instrukcji zależy od jej złożoności i wynosi zazwyczaj od jednego do kilku cykli maszynowych. Model czasowy procesora ma charakter deterministyczny, co oznacza, że liczba cykli wymaganych do wykonania danej instrukcji jest stała i niezależna od kontekstu wykonania.

**Synchronizacja z układem graficznym** Układ graficzny pracuje równolegle z procesorem i cyklicznie przechodzi przez zdefiniowane fazy generowania obrazu. Każda ramka obrazu składa się z sekwencji linii skanowania, a przebieg ich przetwarzania wyznacza rytm pracy całego systemu. W określonych fazach generowania obrazu dostęp procesora do wybranych obszarów pamięci wideo jest ograniczony, co stanowi bezpośrednią konsekwencję współdzielenia zasobów pamięci pomiędzy jednostkami systemu.

**System timerów sprzętowych** Konsola wyposażona jest w sprzętowe układy odmierzania czasu, których działanie jest bezpośrednio powiązane z zegarem systemowym. Timery te generują zdarzenia okresowe wykorzystywane do synchronizacji logiki programu, pomiaru czasu oraz inicjowania przerwań sprzętowych. Ich konfiguracja pozwala na wybór różnych częstotliwości zliczania poprzez dzielenie sygnału zegarowego.

**Znaczenie modelu czasowego** Ścisłe powiązanie pracy wszystkich komponentów z jednym źródłem taktowania zapewnia deterministyczny charakter działania systemu. W konsekwencji po prawność działania oprogramowania zależy nie tylko od sekwencji wykonywanych instrukcji, lecz również od ich dokładnego rozmieszczenia w czasie względem zdarzeń sprzętowych. Model czasowy stanowi zatem istotny element architektury systemowej konsoli i odgrywa kluczową rolę w implementacji wiernych emulatorów platformy.

Procesor zawiera osiem 8-bitowych rejestrów: A,B,C,D,E,F,H,L. Rejestry mogą łączyć się w 16-bitowe rejesty: AF, BC, DE, HL . Rejestr F jest używany jako flagi procesora w wypadku operacji arytmetycznych:

- Bit 7 - Zero (Z), jeśli operacja zwróciła zero
- Bit 6 - Negacja (N), używana gdy ostatnia operacją była porównaniem bądź odejmowaniem
- Bit 5 - Half Carry (H), jeśli doszło do przesunięcia na bicie 3
- Bit 4 - Carry (C), jeśli doszło do przesunięcia na bicie 7

Bity od 3 do 0 są nieużywane i zawsze powinny być zerowane. Rejestr A jest używany jako akumulator w operacjach arytmetycznych.

## 2.3 Zestaw Instrukcji

Zestaw rozkazów (instruction set architecture, ISA) stanowi rozwinięcie koncepcji znanych z mikroprocesorów rodziny Z80, z licznymi uproszczeniami i modyfikacjami dostosowanymi do zastosowań w systemie przenośnym. Wszystkie rozkazy można systematycznie sklasyfikować na poszczególne grupy .

### 2.3.1 Instrukcje transferu danych

Instrukcje tej grupy odpowiadają za przemieszczanie informacji pomiędzy rejestrami, pamięcią oraz natychmiastowymi wartościami liczbowymi. Do najważniejszych operacji należą:

- **LD (Load)** – realizuje kopiowanie danych pomiędzy rejestrami ogólnego przeznaczenia (A, B, C, D, E, H, L), parami rejestrów oraz komórkami pamięci adresowanymi bezpośrednio lub pośrednio.
- **LDH** – specjalizowana forma transferu do i z obszaru pamięci o wysokim adresie (High RAM).
- **PUSH / POP** – zapis i odczyt danych na stosie, z wykorzystaniem wskaźnika stosu SP.

Instrukcje transferu nie modyfikują danych źródłowych (z wyjątkiem operacji stosowych), a ich podstawową funkcją jest reorganizacja informacji w przestrzeni rejestrowo-pamięciowej procesora.

### 2.3.2 Instrukcje arytmetyczne

Instrukcje arytmetyczne realizują operacje matematyczne na liczbach całkowitych. W SM83 operacje te wykonywane są głównie na akumulatorze A. Do kluczowych operacji należą:

- **ADD / ADC** – dodawanie bez i z uwzględnieniem flagi przeniesienia,
- **SUB / SBC** – odejmowanie bez i z uwzględnieniem przeniesienia,
- **INC / DEC** – inkrementacja i dekrementacja rejestrów oraz komórek pamięci,
- **DAA** – korekcja wyniku dodawania do postaci BCD,
- **CP** – porównanie wartości bez zapisu wyniku (modyfikowane są jedynie flagi).

Instrukcje te aktualizują rejestr flag (Z, N, H, C), który odzwierciedla właściwości wyniku operacji.

## Game Boy CPU (SM83) instruction set (JSON)

	x0	x1	x2	x3	x4	x5	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NOP 1 4	LD BC,n16 3 32	LD [BC] A 1 8	INC BC 1 4	DEC B 1 4	LDB,n8 2 8	RLOCA 3 20	LD [a16] SP 3 20	ADD HL, BC 1 8	LD A [BC] 1 8	DEC BC 1 8	INC C 1 4	DEC C 1 4	LD C,n8 2 8	RRCA 1 000 C
1x	STOP,n8 2 4	LD DE,n16 3 32	LD [DE] A 1 8	INC D 1 4	DEC D 1 4	LDD,n8 2 8	RLA 2 12	JR e8 2 12	ADD HL, DE 1 8	LD A [DE] 1 8	DEC DE 1 8	INC E 1 4	DEC E 1 4	LD E,n8 2 8	RRA 1 000 C
2x	JR NZ,n8 2 128	LD HL,n16 3 32	LD [HL] A 1 8	INC H 1 4	DEC H 1 4	LDH,n8 2 8	DAA 2 128	JR Z,e8 2 128	ADD HL, HL 1 8	LD A [HL] 1 8	DEC HL 1 8	INC L 1 4	DEC L 1 4	LD L,n8 2 8	CPL 1 111 C
3x	JR NC,n8 2 128	LD SP,n16 3 32	LD [HL] A 1 8	INC [HL] A 1 8	DEC [HL] A 1 8	LD [HL] A 2 12	JRC,e8 2 128	ADD HL, SP 1 8	LD A [HL] 1 8	DEC SP 1 8	INC A 1 4	DEC A 1 4	LD A,n8 2 8	CCF 1 000 C	
4x	LD B,C 1 4	LD B,D 1 4	LD B,E 1 4	LD B,H 1 4	LD B,L 1 4	LD B, 1 4	LD B,C 1 8	LD A, 1 4	LD C,B 1 4	LD C,D 1 4	LD C,E 1 4	LD C,H 1 4	LD C,I 1 4	LD C,A 1 8	LD C,A 1 4
5x	LD D,B 1 4	LD D,C 1 4	LD D,D 1 4	LD D,E 1 4	LD D,H 1 4	LD D,L 1 4	LD D,C 1 8	LD A, 1 4	LD E,B 1 4	LD E,D 1 4	LD E,E 1 4	LD E,H 1 4	LD E,L 1 4	LD E,A 1 8	LD E,A 1 4
6x	LD H,B 1 4	LD H,C 1 4	LD H,D 1 4	LD H,E 1 4	LD H,H 1 4	LD H,L 1 4	LD H,C 1 8	LD A, 1 4	LD L,B 1 4	LD L,D 1 4	LD L,E 1 4	LD L,H 1 4	LD L,I 1 4	LD L,A 1 8	LD L,A 1 4
7x	LD H[B],L 1 8	LD H[C],L 1 8	LD H[D],L 1 8	LD H[E],L 1 8	LD H[H],L 1 8	HALT 1 4	LD H[A] 1 8	LD A,B 1 4	LD C,D 1 4	LD D,E 1 4	LD E,F 1 4	LD F,G 1 4	LD G,H 1 4	LD A,L 1 8	LD A,A 1 4
8x	ADD A,B 1 4	ADD A,D 1 4	ADD A,E 1 4	ADD A,H 1 4	ADD A,L 1 4	ADD A, 1 4	ADD [HL] A 1 8	ADD A,A 1 4	ADC A,B 1 4	ADC A,C 1 4	ADC A,D 1 4	ADC A,H 1 4	ADC A,L 1 4	ADC A,[HL] 1 8	ADC A,A 1 4
9x	SUB A,B 1 4	SUB A,C 1 4	SUB A,D 1 4	SUB A,E 1 4	SUB A,H 1 4	SUB A, 1 4	SUB [HL] A 1 8	SUB A,A 1 4	SBC A,B 1 4	SBC A,C 1 4	SBC A,D 1 4	SBC A,H 1 4	SBC A,L 1 4	SBC A,[HL] 1 8	SBC A,A 1 4
Ax	AND A,B 1 4	AND A,C 1 4	AND A,D 1 4	AND A,E 1 4	AND A,H 1 4	AND A,L 1 4	AND [HL] A 1 8	AND A,A 1 4	XOR A,B 1 4	XOR A,C 1 4	XOR A,D 1 4	XOR A,H 1 4	XOR A,L 1 4	XOR A,[HL] 1 8	XOR A,A 1 4
Bx	ORA B,A 1 4	ORA C,A 1 4	ORA D,A 1 4	ORA E,A 1 4	ORA H,A 1 4	ORA L,A 1 4	ORA [HL] A 1 8	ORA A,A 1 4	CPA B,C 1 4	CPA D,E 1 4	CPA F,H 1 4	CPA L,H 1 4	CPA [HL] A 1 4	CPA A,A 1 4	
Cx	RET NZ 1 208	POP BC 3 1612	JP NZ,n16 3 1612	JP a16 3 2412	PUSH BC 3 2412	ADD A,n8 1 16	RST 300 1 208	RETZ 1 16	RET 1 16	JP Z,a16 3 2412	PREFIX 1 4	CALL Z,a16 3 2412	CALL 16 3 24	ADC A,n8 1 16	
Dx	RET NC 1 208	POP DE 3 1612	JP NC,n16 3 2412	CALL NC,a16 3 2412	PUSH DE 3 2412	SUB B,A 1 16	RST 310 1 208	RET C 1 16	RET 1 16	JP C,a16 3 2412	CALL C,a16 3 2412	SBC A,n8 1 16	RST 318 1 16		
E	LDH[w],A 1 8	POP HL 1 12	LDH[G],A 1 8	—	PUSH AF 1 16	AND A,n8 1 16	RST 320 1 208	ADD SP,w# 2 16	JP H 1 16	LD [w16],A 1 8	—	—	XOR A,n8 1 16	RST 328 1 16	
Fx	LDH A,[8] 2 12	POP AF 1 12	LDH A,[C] 1 8	DI 1 4	PUSH AF 1 16	OR A,n8 1 16	RST 330 2 008	LD H,SP+w# 2 12	LD SP, H 0 00C	LD A,[w16] 1 16	EI 1 4	—	CP A,n8 2 8	RST 338 2 16	

Rysunek 1: Tablica rozmieszczenia instrukcji: [3]

## Prefixed (\$CB \$xx)

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	RLC,B 2 8	RLG,D 2 8	RLG,E 2 8	RLC,H 2 8	RLC,L 2 8	RLC, 2 8	RLC,[HL] 2 16	RLC,A 2 8	RLC,B 2 8	RLC,C 2 8	RLC,D 2 8	RLC,E 2 8	RLC,H 2 8	RLC,[HL] 2 16	RRCA 2 8	
1x	RLC,B 2 8	RLG,D 2 8	RLG,E 2 8	RLC,H 2 8	RLC,L 2 8	RLC, 2 8	RLC,[HL] 2 16	RLC,A 2 8	RLR,B 2 8	RLR,C 2 8	RLR,D 2 8	RLR,E 2 8	RLR,H 2 8	RLR,[HL] 2 16	RRA 2 00 C	
2x	SLA,B 2 8	SLA,D 2 8	SLA,E 2 8	SLA,H 2 8	SLA,L 2 8	SLA, 2 8	SLA,[HL] 2 16	SLA,A 2 8	SLA,B 2 8	SLA,C 2 8	SLA,D 2 8	SLA,E 2 8	SLA,H 2 8	SLA,[HL] 2 16	SPA,A 2 8	
3x	SHAD,B 2 8	SHAD,D 2 8	SHAD,E 2 8	SHAD,H 2 8	SHAD,L 2 8	SHAD, 2 8	SHAD,[HL] 2 16	SHAD,A 2 8	SHR,B 2 8	SHR,C 2 8	SHR,D 2 8	SHR,E 2 8	SHR,H 2 8	SHR,[HL] 2 16	SPD,A 2 8	
4x	BIT0,B 2 8	BIT0,C 2 8	BIT0,D 2 8	BIT0,E 2 8	BIT0,H 2 8	BIT0, 2 8	BIT0,[HL] 2 16	BIT0,A 2 8	BIT1,B 2 8	BIT1,C 2 8	BIT1,D 2 8	BIT1,E 2 8	BIT1,H 2 8	BIT1,[HL] 2 16	BIT1,A 2 8	
5x	BIT2,B 2 8	BIT2,C 2 8	BIT2,D 2 8	BIT2,E 2 8	BIT2,H 2 8	BIT2, 2 8	BIT2,[HL] 2 16	BIT2,A 2 8	BIT3,B 2 8	BIT3,C 2 8	BIT3,D 2 8	BIT3,E 2 8	BIT3,H 2 8	BIT3,[HL] 2 16	BIT3,A 2 8	
6x	BIT4,B 2 8	BIT4,C 2 8	BIT4,D 2 8	BIT4,E 2 8	BIT4,H 2 8	BIT4, 2 8	BIT4,[HL] 2 16	BIT4,A 2 8	BIT5,B 2 8	BIT5,C 2 8	BIT5,D 2 8	BIT5,E 2 8	BIT5,H 2 8	BIT5,[HL] 2 16	BIT5,A 2 8	
7x	BIT6,B 2 8	BIT6,C 2 8	BIT6,D 2 8	BIT6,E 2 8	BIT6,H 2 8	BIT6, 2 8	BIT6,[HL] 2 16	BIT6,A 2 8	BIT7,B 2 8	BIT7,C 2 8	BIT7,D 2 8	BIT7,E 2 8	BIT7,H 2 8	BIT7,[HL] 2 16	BIT7,A 2 8	
8x	RES0,B 2 8	RES0,C 2 8	RES0,D 2 8	RES0,E 2 8	RES0,H 2 8	RES0,L 2 8	RES0,[HL] 2 16	RES0,A 2 8	RES1,B 2 8	RES1,C 2 8	RES1,D 2 8	RES1,E 2 8	RES1,H 2 8	RES1,[HL] 2 16	RES1,A 2 8	
9x	RES2,B 2 8	RES2,C 2 8	RES2,D 2 8	RES2,E 2 8	RES2,H 2 8	RES2,L 2 8	RES2,[HL] 2 16	RES2,A 2 8	RES3,B 2 8	RES3,C 2 8	RES3,D 2 8	RES3,E 2 8	RES3,H 2 8	RES3,[HL] 2 16	RES3,A 2 8	
Ax	RES4,B 2 8	RES4,C 2 8	RES4,D 2 8	RES4,E 2 8	RES4,H 2 8	RES4,L 2 8	RES4,[HL] 2 16	RES4,A 2 8	RES5,B 2 8	RES5,C 2 8	RES5,D 2 8	RES5,E 2 8	RES5,H 2 8	RES5,[HL] 2 16	RES5,A 2 8	
Bx	RES6,B 2 8	RES6,C 2 8	RES6,D 2 8	RES6,E 2 8	RES6,H 2 8	RES6,L 2 8	RES6,[HL] 2 16	RES6,A 2 8	RES7,B 2 8	RES7,C 2 8	RES7,D 2 8	RES7,E 2 8	RES7,H 2 8	RES7,[HL] 2 16	RES7,A 2 8	
Cx	SET0,B 2 8	SET0,C 2 8	SET0,D 2 8	SET0,E 2 8	SET0,H 2 8	SET0,L 2 8	SET0,[HL] 2 16	SET0,A 2 8	SET1,B 2 8	SET1,C 2 8	SET1,D 2 8	SET1,E 2 8	SET1,H 2 8	SET1,[HL] 2 16	SET1,A 2 8	
Dx	SET2,B 2 8	SET2,C 2 8	SET2,D 2 8	SET2,E 2 8	SET2,H 2 8	SET2,L 2 8	SET2,[HL] 2 16	SET2,A 2 8	SET3,B 2 8	SET3,C 2 8	SET3,D 2 8	SET3,E 2 8	SET3,H 2 8	SET3,[HL] 2 16	SET3,A 2 8	
E	SET4,B 2 8	SET4,C 2 8	SET4,D 2 8	SET4,E 2 8	SET4,H 2 8	SET4,L 2 8	SET4,[HL] 2 16	SET4,A 2 8	SET5,B 2 8	SET5,C 2 8	SET5,D 2 8	SET5,E 2 8	SET5,H 2 8	SET5,[HL] 2 16	SET5,A 2 8	
Fx	SET6,B 2 8	SET6,C 2 8	SET6,D 2 8	SET6,E 2 8	SET6,H 2 8	SET6,L 2 8	SET6,[HL] 2 16	SET6,A 2 8	SET7,B 2 8	SET7,C 2 8	SET7,D 2 8	SET7,E 2 8	SET7,H 2 8	SET7,[HL] 2 16	SET7,A 2 8	

Rysunek 2: Tablica rozmieszczenia instrukcji po opcjedzie 0xCB: [3]

### 2.3.3 Instrukcje logiczne i bitowe

Grupa ta obejmuje operacje algebry Boole'a oraz manipulacje pojedynczymi bitami. Do najważniejszych należą:

- **AND, OR, XOR** – operacje logiczne na akumulatorze,
- **CPL** – negacja bitowa zawartości akumulatora,
- **BIT** – test wybranego bitu,
- **SET / RES** – ustawienie lub wyzerowanie wskazanego bitu.

Instrukcje bitowe umożliwiają precyzyjną kontrolę struktur danych, co jest szczególnie istotne w systemach o ograniczonych zasobach pamięci.

### 2.3.4 Instrukcje przesunięć i rotacji

Operacje tej kategorii realizują przesunięcia bitowe oraz rotacje z lub bez udziału flagi przeniesienia:

- **RL, RLC** – rotacja w lewo,
- **RR, RRC** – rotacja w prawo,
- **SLA, SRA, SRL** – przesunięcia arytmetyczne i logiczne,
- **SWAP** – zamiana pół bajtów w rejestrze.

Instrukcje te są użyteczne zarówno w obliczeniach arytmetycznych, jak i w przetwarzaniu danych binarnych.

### 2.3.5 Instrukcje sterowania przepływem programu

Instrukcje sterujące odpowiadają za zmianę kolejności wykonywania programu:

- **JP** – skok bezwarunkowy lub warunkowy,
- **JR** – skok względny o ograniczonym zasięgu,
- **CALL / RET** – wywołanie i powrót z podprogramu,
- **RST** – szybkie wywołanie procedury o stałym adresie.

Warunkowość operacji zależy od stanu flag procesora, co umożliwia implementację struktur decyzyjnych.

Do tej grupy należą instrukcje wpływające na stan globalny procesora:

- **NOP** – brak operacji,
- **HALT** – zatrzymanie pracy CPU do momentu wystąpienia przerwania,
- **STOP** – tryb niskiego poboru energii,
- **DI / EI** – dezaktywacja i aktywacja systemu przerwań,
- **SCF / CCF** – manipulacja flagą przeniesienia.

Instrukcje te mają kluczowe znaczenie dla zarządzania energią oraz obsługi zdarzeń asynchronicznych.

## 2.4 Mapowanie pamięci w konsoli Game Boy

Architektura pamięci konsoli Game Boy oparta jest na 16-bitowej przestrzeni adresowej o rozmiarze  $2^{16} = 65\,536$  bajtów (64 KB). Procesor systemu, zgodny z modyfikowaną architekturą Sharp LR35902, wykorzystuje jednolitą przestrzeń adresową, w której różne zakresy adresów są przypisane do odmiennych typów pamięci oraz rejestrów sprzętowych. Taki model organizacji pamięci określany jest jako *memory-mapped I/O* i umożliwia bezpośrednią komunikację procesora z układami peryferyjnymi poprzez operacje odczytu i zapisu w odpowiednich obszarach adresowych.

Zakres adresów	Rozmiar	Przeznaczenie
0000-3FFF	16 KB	Stała pamięć ROM banku 0
4000-7FFF	16 KB	Przełączalny bank ROM
8000-9FFF	8 KB	Pamięć wideo (VRAM)
A000-BFFF	8 KB	Zewnętrzna pamięć RAM w kartridżu
C000-CFFF	4 KB	Wewnętrzna pamięć robocza (WRAM bank 0)
D000-DFFF	4 KB	Wewnętrzna pamięć robocza (bank przełączalny)
E000-FDFF	7.5 KB	Obszar echo pamięci WRAM
FE00-FE9F	160 B	Pamięć atrybutów obiektów (OAM)
FEAO-FEFF	96 B	Obszar niedostępny
FF00-FF7F	128 B	Rejestry wejścia/wyjścia
FF80-FFFE	127 B	Pamięć szybka (HRAM)
FFFF	1 B	Rejestr przerwań (IE)

Tabela 1: Mapa pamięci konsoli Game Boy [5]

#### 2.4.1 Ogólny układ przestrzeni adresowej

Przestrzeń adresowa Game Boya podzielona jest na segmenty o ustalonej funkcji sprzętowej. Tabela 1 przedstawia standardowy schemat mapowania pamięci.

#### 2.4.2 Pamięć programu i bankowanie ROM

Dolne 32 KB przestrzeni adresowej przeznaczone jest na pamięć tylko do odczytu (ROM) zawartą w kartridżu. Pierwsze 16 KB (0000-3FFF) stanowi bank stały, natomiast zakres 4000-7FFF obsługuje mechanizm przełączania banków pamięci, realizowany przez kontroler MBC (Memory Bank Controller). Dzięki temu możliwe jest adresowanie programów o rozmiarze znacznie przekraczającym 64 KB.

#### 2.4.3 Pamięć wideo

Obszar 8000-9FFF odpowiada pamięci VRAM, wykorzystywanej przez układ graficzny do przechowywania danych kafelków graficznych oraz map tła. Dostęp do tej pamięci jest ograniczony w określonych fazach cyklu wyświetlania obrazu, co wynika z synchronizacji procesora z kontrolerem LCD.

#### 2.4.4 Pamięć robocza i obszar echo

Wewnętrzna pamięć robocza WRAM zajmuje zakres C000-DFFF. Dodatkowo obszar E000-FDFF stanowi tzw. echo WRAM, czyli lustrzane odwzorowanie części pamięci roboczej. Jego obecność wynika z uproszczeń sprzętowych i nie jest zalecana do programowego wykorzystania.

#### 2.4.5 Mapowanie urządzeń wejścia/wyjścia

Zakres FF00-FF7F zawiera rejesty sterujące sprzętem, w tym kontrolerem dźwięku, systemem przerwań, timerami oraz interfejsem wejściowym. Operacje zapisu i odczytu w tych adresach odpowiadają bezpośrednim operacjom na urządzeniach peryferyjnych.

#### 2.4.6 Pamięć wysokiej prędkości

Obszar FF80-FFFE, określany jako HRAM (High RAM), to niewielki fragment szybkiej pamięci roboczej o krótkim czasie dostępu. Ze względu na swoje właściwości jest on często wykorzystywany do przechowywania zmiennych o krytycznym znaczeniu czasowym.

#### 2.4.7 Znaczenie architektury mapowania pamięci

Przyjęty model mapowania pamięci umożliwia efektywne współdzielenie przestrzeni adresowej przez kod programu, dane oraz układy sprzętowe. Rozwiążanie to upraszcza konstrukcję systemu, lecz jednocześnie wymaga ścisłej kontroli dostępu do poszczególnych segmentów pamięci przez oprogramowanie systemowe i aplikacyjne.

## 2.5 System przerwań w konsoli Game Boy

System przerwań w konsoli Game Boy realizuje mechanizm asynchronicznej obsługi zdarzeń sprzętowych, umożliwiając procesorowi reagowanie na zdarzenia wewnętrzne i zewnętrzne bez konieczności ciągłego odpytywania urządzeń peryferyjnych. Architektura ta stanowi kluczowy element synchronizacji pracy jednostki centralnej z układami graficznymi, licznikami czasu oraz interfejsem wejściowym.

### 2.5.1 Źródła przerwań

Konsola Game Boy obsługuje pięć podstawowych źródeł przerwań sprzętowych. Każdemu z nich przypisany jest odrębny wektor przerwania oraz odpowiedni bit w rejestrach sterujących. Zestaw źródeł przerwań przedstawiono w tabeli 2.

Bit	Nazwa przerwania	Funkcja
0	V-Blank	Zakończenie rysowania ramki obrazu
1	LCD STAT	Zdarzenia kontrolera LCD
2	Timer	Przepełnienie timera systemowego
3	Serial	Zakończenie transmisji szeregowej
4	Joypad	Zmiana stanu kontrolera wejściowego

Tabela 2: Źródła przerwań w konsoli Game Boy

Przerwanie V-Blank pełni szczególną rolę, ponieważ wyznacza bezpieczny moment aktualizacji pamięci wideo. Przerwanie LCD STAT umożliwia reakcję na szczegółowe stany pracy kontrolera wyświetlania, natomiast przerwanie timera wykorzystywane jest do odmierzania czasu oraz synchronizacji logiki programu.

### 2.5.2 Rejestry sterujące systemem przerwań

Obsługa przerwań realizowana jest poprzez dwa rejesty mapowane w przestrzeni adresowej:

- IE (Interrupt Enable, adres FFFF) — określa, które źródła przerwań są aktywne.
- IF (Interrupt Flag, adres FF0F) — przechowuje informację o zgłoszonych przerwaniach.

Każdy bit w obu rejestrach odpowiada jednemu źródłu przerwania. Zgłoszenie przerwania polega na ustawieniu odpowiedniego bitu w rejestrze IF. Jeżeli odpowiadający mu bit w rejestrze IE jest ustawiony oraz globalna obsługa przerwań jest włączona, procesor inicjuje procedurę obsługi przerwania.

### 2.5.3 Mechanizm obsługi przerwania

Po wykryciu aktywnego przerwania procesor wykonuje następującą sekwencję działań:

1. zakończenie bieżącej instrukcji,
2. zapis aktualnej wartości licznika programu na stosie,
3. wyłączenie globalnej obsługi przerwań,
4. skok pod adres wektora przerwania.

Adresy wektorów przerwań są stałe i przypisane do konkretnych źródeł zdarzeń. Po zakończeniu procedury obsługi przerwania wykonywana jest instrukcja powrotu, która przywraca poprzedni stan wykonania programu.

### 2.5.4 Priorytety przerwań

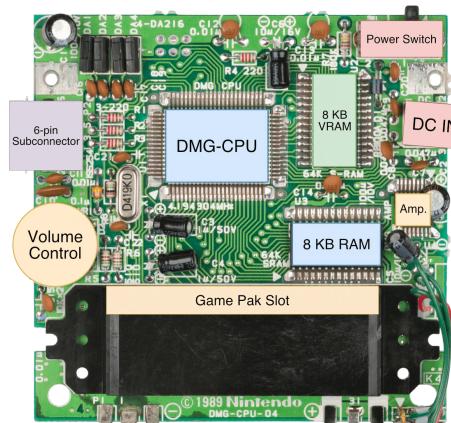
W przypadku jednoczesnego zgłoszenia wielu przerwań stosowany jest ustalony porządek priorytetów. Najwyższy priorytet posiada przerwanie V-Blank, a najniższy przerwanie kontrolera wejściowego. Taki schemat zapewnia deterministyczną obsługę zdarzeń związanych z generowaniem obrazu, które są krytyczne dla poprawnego działania systemu.

### 2.5.5 Rola systemu przerwań w architekturze konsoli

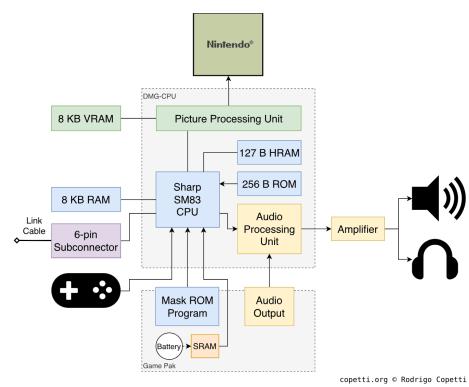
System przerwań stanowi podstawowy mechanizm synchronizacji komponentów sprzętowych konsoli Game Boy. Dzięki niemu możliwe jest efektywne współdziałanie procesora z kontrolerem graficznym, licznikami czasu oraz urządzeniami wejścia/wyjścia przy zachowaniu deterministycznego modelu wykonania programu. Mechanizm ten ma istotne znaczenie zarówno dla projektowania oprogramowania systemowego, jak i dla implementacji emulatorów platformy.

## 2.6 Płyta główna

Płyta główna konsoli Game Boy stanowi podstawowy element konstrukcyjny systemu, integrujący wszystkie kluczowe układy elektroniczne odpowiedzialne za przetwarzanie danych, generowanie obrazu, obsługę dźwięku oraz zarządzanie energią. Ze względu na przenośny charakter urządzenia, konstrukcja płyty została zaprojektowana z naciskiem na energooszczędność, prostotę architektury oraz wysoką niezawodność.



Rysunek 3: Zdjęcie płyty głównej oryginalnego GameBoy'a z opisami komponentów



Rysunek 4: Schemat płyty głównej konsoli GameBoy

Rysunek 5: Źródło: [1]

### 2.6.1 Jednostka centralna

Centralnym elementem płyty głównej jest układ DMG-CPU, stanowiący zintegrowany system typu System on Chip (SoC). W przeciwieństwie do klasycznych konstrukcji mikrokomputerowych, w których poszczególne funkcje realizowane są przez oddzielne układy scalone, DMG-CPU integruje w jednej strukturze półprzewodnikowej jednostkę obliczeniową, kontroler pamięci, układ graficzny, generator dźwięku oraz logikę sterującą systemem.

Rdzeń obliczeniowy układu bazuje na architekturze 8-bitowej z 16-bitową przestrzenią adresową i realizuje wykonywanie programu oraz zarządzanie przepływem danych pomiędzy komponentami systemu. Integracja funkcji w jednym układzie pozwoliła na ograniczenie liczby połączeń na płycie głównej, zmniejszenie poboru energii oraz zwiększenie niezawodności urządzenia.

DMG-CPU odpowiada również za synchronizację pracy podsystemów, obsługę przerwań sprzętowych, sterowanie dostępem do pamięci operacyjnej i wideo oraz koordynację komunikacji z interfejsem kartridżu i urządzeniami wejścia-wyjścia. Zastosowanie architektury SoC stanowiło kluczowy element miniaturyzacji konsoli oraz umożliwiło osiągnięcie wysokiej efektywności energetycznej przy zachowaniu pełnej funkcjonalności systemu.

### 2.6.2 Pamięć operacyjna i pamięć wideo

Na płycie głównej znajdują się układy pamięci pełniące różne funkcje w architekturze systemu:

- **WRAM (Work RAM)** – pamięć operacyjna wykorzystywana przez procesor do przechowywania danych roboczych oraz stosu.
  - **VRAM (Video RAM)** – pamięć przeznaczona do przechowywania danych graficznych, takich jak mapy kafli, wzorce znaków oraz informacje o sprite'ach.

Pamięci te współpracują bezpośrednio z procesorem oraz układem generowania obrazu, zapewniając sprawną wymianę danych niezbędnych do działania systemu.

#### **2.6.3 Układ graficzny**

Funkcje generowania obrazu realizowane są przez zintegrowany kontroler LCD, który odpowiada za przetwarzanie danych graficznych i sterowanie wyświetlaczem ciekłokrystalicznym. Układ ten obsługuje system kafli (tile-based graphics), renderowanie sprite'ów oraz synchronizację obrazu z odświeżaniem ekranu. Kontroler LCD współpracuje bezpośrednio z pamięcią wideo oraz procesorem, tworząc kompletny tor przetwarzania grafiki.

#### **2.6.4 Układ dźwiękowy**

System dźwiękowy konsoli jest zintegrowany z architekturą płyty głównej i składa się z generatorów fal dźwiękowych sterowanych programowo. Układ ten umożliwia generowanie kilku kanałów audio, w tym fal prostokątnych, szumu oraz próbek o ograniczonej rozdzielczości. Sygnał audio kierowany jest następnie do wzmacniacza i głośnika lub wyjścia słuchawkowego. Ze względu skomplikowania, komponent układu dźwiękowego nie będzie uwzględniany.

#### **2.6.5 Interfejs kartridża**

Na płycie głównej znajduje się złącze kartridża umożliwiające komunikację z zewnętrznym nośnikiem oprogramowania. Interfejs ten zapewnia:

- dostęp do pamięci programu,
- komunikację z dodatkowymi układami w kartridżu,
- możliwość rozszerzenia funkcjonalności systemu.

Zastosowanie wymiennych kartridży umożliwiło modularną architekturę systemu oraz łatwą dystrybucję oprogramowania.

#### **2.6.6 Układy wejścia i wyjścia**

Płyta główna integruje również komponenty odpowiedzialne za komunikację z użytkownikiem oraz otoczeniem:

- kontroler przycisków sterujących,
- interfejs komunikacyjny (port szeregowy),
- sterowanie wyświetlaczem LCD,
- wzmacniacz audio.

Układy te zapewniają interakcję użytkownika z systemem oraz obsługę sygnałów wejściowych i wyjściowych.

Płyta główna konsoli Game Boy stanowi przykład zwartej i funkcjonalnej integracji elementów elektronicznych w systemie o ograniczonych zasobach sprzętowych. Zastosowanie zintegrowanych układów scalonych, zoptymalizowane zarządzanie energią oraz modułowa struktura pamięci i interfejsów umożliwiły stworzenie wydajnego, przenośnego systemu rozrywkowego. Analiza budowy płyty głównej pozwala dostrzec kompromis pomiędzy funkcjonalnością, energooszczędnością a prostotą konstrukcji, charakterystyczny dla projektowania urządzeń przenośnych końca XX wieku.

### **2.7 Kartridże systemu Game Boy**

Kartridże stanowią podstawowy nośnik oprogramowania dla konsoli Game Boy oraz element rozbudowujący funkcjonalność systemu. Oprócz pamięci programu zawierają one układy logiczne odpowiedzialne za zarządzanie pamięcią, zapisywanie danych użytkownika oraz, w niektórych przypadkach, dodatkowe funkcje sprzętowe. Różnorodność konstrukcji kartridżów wynikała z konieczności obsługi gier o rosnącej złożoności przy zachowaniu ograniczeń architektury systemu.

### **2.7.1 Kartridże bez kontrolera banków pamięci (ROM Only)**

Najprostszy typ kartridża zawiera wyłącznie pamięć ROM z programem gry. Konstrukcja ta nie umożliwia przełączania banków pamięci ani zapisu danych użytkownika. Ze względu na ograniczoną pojemność stosowana była głównie w prostszych produkcjach we wcześnieym okresie rozwoju systemu.

### **2.7.2 Kartridże z kontrolerem MBC1**

Kontroler Memory Bank Controller 1 (MBC1) umożliwia przełączanie banków pamięci ROM oraz, opcjonalnie, banków pamięci RAM. Układ ten pozwalał na znaczące zwiększenie maksymalnego rozmiaru programu. Kartridże tego typu występowały w kilku wariantach:

- MBC1 + RAM,
- MBC1 + RAM + bateria podtrzymująca zapis danych.

### **2.7.3 Kartridże z kontrolerem MBC2**

MBC2 integruje funkcję przełączania banków ROM z niewielką, wbudowaną pamięcią RAM przeznaczoną do zapisu danych gry. W odróżnieniu od innych kontrolerów, pamięć RAM jest integralną częścią układu i nie występuje jako oddzielny komponent. Kartridże te standardowo wyposażone były w baterię podtrzymującą zapis.

### **2.7.4 Kartridże z kontrolerem MBC3**

MBC3 rozszerza funkcjonalność poprzednich kontrolerów o zegar czasu rzeczywistego (RTC). Rozwiążanie to umożliwiało implementację mechanik zależnych od upływu czasu rzeczywistego. Warianty konstrukcyjne obejmowały:

- MBC3 + RAM,
- MBC3 + RAM + bateria,
- MBC3 + RAM + bateria + RTC.

### **2.7.5 Kartridże z kontrolerem MBC5**

MBC5 wprowadza rozszerzony system adresowania pamięci ROM oraz obsługę większych pojemności pamięci RAM. Układ ten był stosowany w późniejszych produkcjach o dużej objętości danych. Warianty obejmowały:

- MBC5 + RAM,
- MBC5 + RAM + bateria,
- MBC5 + RAM + bateria + silnik wibracyjny (Rumble).

### **2.7.6 Kartridże specjalizowane**

Oprócz standardowych kontrolerów banków pamięci produkowano kartridże wyposażone w dodatkowe układy funkcjonalne rozszerzające możliwości systemu. Do najważniejszych należą:

- kartridże z czujnikiem ruchu,
- kartridże z pamięcią EEPROM,
- kartridże z dodatkowymi układami logicznymi sterującymi akcesoriami,
- kartridże diagnostyczne i testowe wykorzystywane w procesie produkcyjnym.

### **2.7.7 Struktura fizyczna kartridża**

Typowy kartridż składa się z płytki drukowanej zawierającej układ ROM, opcjonalną pamięć RAM, kontroler banków pamięci oraz element podtrzymywania zasilania pamięci zapisu. Komunikacja z konsolą odbywa się za pośrednictwem złącza krawędziowego, które zapewnia dostęp do magistrali adresowej, magistrali danych oraz linii sterujących.

## Bibliografia

- [1] Rodrigo Copetti. *Game Boy Architecture - A Practical Analysis*. URL: <https://www.copetti.org/writings/consoles/game-boy/> (term. wiz. 15.02.2025).
- [2] *Game Boy*. Wikipedia. URL: [https://en.wikipedia.org/wiki/Game\\_Boy](https://en.wikipedia.org/wiki/Game_Boy) (term. wiz. 16.02.2025).
- [3] *Game Boy Opcode Tables*. URL: <https://gbdev.io/gb-opcodes/optables/> (term. wiz. 15.02.2025).
- [4] Gekkio. *Game Boy: Complete Technical Reference*. URL: <https://gekkio.fi/files/gb-docs/gbctr.pdf> (term. wiz. 15.02.2025).
- [5] *Pan Docs - Game Boy Technical Reference*. URL: <https://gbdev.io/pandocs/> (term. wiz. 15.02.2025).