Niskopoziomowa emulacja sprzętowa implementacja architekturę systemową konsoli Game Boy

Patryk Michalak

20 lutego 2026

Streszczenie

Your abstract.

Spis treści

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 odniosła 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 kartridży (Game Pak), zawierających pamięć ROM z kodem programu, danymi graficznymi oraz dźwiękowymi. W niektórych przypadkach kartridże 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 rozdzielczości 160×144 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 kartridża 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 rozdzielczości 160x144 pikseli, D-padem, czterema przyciskami gry i jednym głośnikiem. Używa stworzonych na potrzebe konsoli kartridże '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 wytworzyny na potrzeby konsoli 8-bitowy procesor SM83 firmy Sharp. [4]. Procesor był mieszanką 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 płytka 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 kompomentami 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 poprawność 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 rejestry: AF, BC, DE, HL . Rejestr F jest używany jako flagi procesora w wypadku operacji arymetycznych:

- 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 operacja arytmetcznych.

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. Wszytkie 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	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NOP 1 4 ----	LD BC, n16 3 12 ----	LD [BC], A 1 8 ----	INC BC 1 8 ----	INC B 1 4 Z0H-	DEC B 1 4 Z1H-	LD B, n8 2 8 ----	RLCA 1 4 000C	LD [a16], SP 3 20 ----	ADD HL, BC 1 8 -0HC	LD A, [BC] 1 8 ----	DEC BC 1 8 ----	INC C 1 4 Z0H-	DEC C 1 4 Z1H-	LD C, n8 2 8 ----	RRCA 1 4 000C
1x	STOP n8 2 4 ----	LD DE, n16 3 12 ----	LD [DE], A 1 8 ----	INC DE 1 8 ----	INC D 1 4 Z0H-	DEC D 1 4 Z1H-	LD D, n8 2 8 ----	RLA 1 4 000C	JR e8 2 12 ----	ADD HL, DE 1 8 -0HC	LD A, [DE] 1 8 ----	DEC DE 1 8 ----	INC E 1 4 Z0H-	DEC E 1 4 Z1H-	LD E, n8 2 8 ----	RRA 1 4 000C
2x	JR NZ, e8 2 12/8 ----	LD HL, n16 3 12 ----	LD [HL+], A 1 8 ----	INC HL 1 8 ----	INC H 1 4 Z0H-	DEC H 1 4 Z1H-	LD H, n8 2 8 ----	DAA 1 4 Z-0C	JR Z, e8 2 12/8 ----	ADD HL, HL 1 8 -0HC	LD A, [HL+] 1 8 ----	DEC HL 1 8 ----	INC L 1 4 Z0H-	DEC L 1 4 Z1H-	LD L, n8 2 8 ----	CPL 1 4 -11-
3x	JR NC, e8 2 12/8 ----	LD SP, n16 3 12 ----	LD [HL-], A 1 8 ----	INC SP 1 8 ----	INC [HL] 1 12 Z0H-	DEC [HL] 1 12 Z1H-	LD [HL], n8 2 12 ----	SCF 1 4 -001	JR C, e8 2 12/8 ----	ADD HL, SP 1 8 -0HC	LD A, [HL-] 1 8 ----	DEC SP 1 8 ----	INC A 1 4 Z0H-	DEC A 1 4 Z1H-	LD A, n8 2 8 ----	CCF 1 4 -00C
4x	LD B, B 1 4 ----	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, [HL] 1 8 ----	LD B, A 1 4 ----	LD C, B 1 4 ----	LD C, C 1 4 ----	LD C, D 1 4 ----	LD C, E 1 4 ----	LD C, H 1 4 ----	LD C, L 1 4 ----	LD C, [HL] 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, [HL] 1 8 ----	LD D, A 1 4 ----	LD E, B 1 4 ----	LD E, C 1 4 ----	LD E, D 1 4 ----	LD E, E 1 4 ----	LD E, H 1 4 ----	LD E, L 1 4 ----	LD E, [HL] 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, [HL] 1 8 ----	LD H, A 1 4 ----	LD L, B 1 4 ----	LD L, C 1 4 ----	LD L, D 1 4 ----	LD L, E 1 4 ----	LD L, H 1 4 ----	LD L, L 1 4 ----	LD L, [HL] 1 8 ----	LD L, A 1 4 ----
7x	LD [HL], B 1 8 ----	LD [HL], C 1 8 ----	LD [HL], D 1 8 ----	LD [HL], E 1 8 ----	LD [HL], H 1 8 ----	LD [HL], L 1 8 ----	HALT 1 4 ----	LD [HL], A 1 8 ----	LD A, B 1 4 ----	LD A, C 1 4 ----	LD A, D 1 4 ----	LD A, E 1 4 ----	LD A, H 1 4 ----	LD A, L 1 4 ----	LD A, [HL] 1 8 ----	LD A, A 1 4 ----
8x	ADD A, B 1 4 Z0HC	ADD A, C 1 4 Z0HC	ADD A, D 1 4 Z0HC	ADD A, E 1 4 Z0HC	ADD A, H 1 4 Z0HC	ADD A, L 1 4 Z0HC	ADD A, [HL] 1 8 Z0HC	ADD A, A 1 4 Z0HC	ADC A, B 1 4 Z0HC	ADC A, C 1 4 Z0HC	ADC A, D 1 4 Z0HC	ADC A, E 1 4 Z0HC	ADC A, H 1 4 Z0HC	ADC A, L 1 4 Z0HC	ADC A, [HL] 1 8 Z0HC	ADC A, A 1 4 Z0HC
9x	SUB A, B 1 4 Z1HC	SUB A, C 1 4 Z1HC	SUB A, D 1 4 Z1HC	SUB A, E 1 4 Z1HC	SUB A, H 1 4 Z1HC	SUB A, L 1 4 Z1HC	SUB A, [HL] 1 8 Z1HC	SUB A, A 1 4 1100	SBC A, B 1 4 Z1HC	SBC A, C 1 4 Z1HC	SBC A, D 1 4 Z1HC	SBC A, E 1 4 Z1HC	SBC A, H 1 4 Z1HC	SBC A, L 1 4 Z1HC	SBC A, [HL] 1 8 Z1HC	SBC A, A 1 4 Z1H-
Ax	AND A, B 1 4 Z010	AND A, C 1 4 Z010	AND A, D 1 4 Z010	AND A, E 1 4 Z010	AND A, H 1 4 Z010	AND A, L 1 4 Z010	AND A, [HL] 1 8 Z010	AND A, A 1 4 Z010	XOR A, B 1 4 Z000	XOR A, C 1 4 Z000	XOR A, D 1 4 Z000	XOR A, E 1 4 Z000	XOR A, H 1 4 Z000	XOR A, L 1 4 Z000	XOR A, [HL] 1 8 Z000	XOR A, A 1 4 1000
Bx	OR A, B 1 4 Z000	OR A, C 1 4 Z000	OR A, D 1 4 Z000	OR A, E 1 4 Z000	OR A, H 1 4 Z000	OR A, L 1 4 Z000	OR A, [HL] 1 8 Z000	OR A, A 1 4 Z000	CP A, B 1 4 Z1HC	CP A, C 1 4 Z1HC	CP A, D 1 4 Z1HC	CP A, E 1 4 Z1HC	CP A, H 1 4 Z1HC	CP A, L 1 4 Z1HC	CP A, [HL] 1 8 Z1HC	CP A, A 1 4 1100
Cx	RET NZ 1 20/8 ----	POP BC 1 12 ----	JP NZ, a16 3 16/12 ----	JP a16 3 16 ----	CALL NZ, a16 3 24/12 ----	PUSH BC 1 16 ----	ADD A, n8 2 8 Z0HC	RST $00 1 16 ----	RET Z 1 20/8 ----	RET 1 16 ----	JP Z, a16 3 16/12 ----	PREFIX 1 4 ----	CALL Z, a16 3 24/12 ----	CALL a16 3 24 ----	ADC A, n8 2 8 Z0HC	RST $08 1 16 ----
Dx	RET NC 1 20/8 ----	POP DE 1 12 ----	JP NC, a16 3 16/12 ----	—	CALL NC, a16 3 24/12 ----	PUSH DE 1 16 ----	SUB A, n8 2 8 Z1HC	RST $10 1 16 ----	RET C 1 20/8 ----	RETI 1 16 ----	JP C, a16 3 16/12 ----	—	CALL C, a16 3 24/12 ----	—	SBC A, n8 2 8 Z1HC	RST $18 1 16 ----
Ex	LDH [a8], A 2 12 ----	POP HL 1 12 ----	LDH [C], A 1 8 ----	—	—	PUSH HL 1 16 ----	AND A, n8 2 8 Z010	RST $20 1 16 ----	ADD SP, e8 2 16 00HC	JP HL 1 4 ----	LD [a16], A 3 16 ----	—	—	—	XOR A, n8 2 8 Z000	RST $28 1 16 ----
Fx	LDH A, [a8] 2 12 ----	POP AF 1 12 ZNHC	LDH A, [C] 1 8 ----	DI 1 4 ----	—	PUSH AF 1 16 ----	OR A, n8 2 8 Z000	RST $30 1 16 ----	LD HL, SP + e8 2 12 00HC	LD SP, HL 1 8 ----	LD A, [a16] 3 16 ----	EI 1 4 ----	—	—	CP A, n8 2 8 Z1HC	RST $38 1 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 Z00C	RLC C 2 8 Z00C	RLC D 2 8 Z00C	RLC E 2 8 Z00C	RLC H 2 8 Z00C	RLC L 2 8 Z00C	RLC [HL] 2 16 Z00C	RLC A 2 8 Z00C	RRC B 2 8 Z00C	RRC C 2 8 Z00C	RRC D 2 8 Z00C	RRC E 2 8 Z00C	RRC H 2 8 Z00C	RRC L 2 8 Z00C	RRC [HL] 2 16 Z00C	RRC A 2 8 Z00C
1x	RL B 2 8 Z00C	RL C 2 8 Z00C	RL D 2 8 Z00C	RL E 2 8 Z00C	RL H 2 8 Z00C	RL L 2 8 Z00C	RL [HL] 2 16 Z00C	RL A 2 8 Z00C	RR B 2 8 Z00C	RR C 2 8 Z00C	RR D 2 8 Z00C	RR E 2 8 Z00C	RR H 2 8 Z00C	RR L 2 8 Z00C	RR [HL] 2 16 Z00C	RR A 2 8 Z00C
2x	SLA B 2 8 Z00C	SLA C 2 8 Z00C	SLA D 2 8 Z00C	SLA E 2 8 Z00C	SLA H 2 8 Z00C	SLA L 2 8 Z00C	SLA [HL] 2 16 Z00C	SLA A 2 8 Z00C	SRA B 2 8 Z00C	SRA C 2 8 Z00C	SRA D 2 8 Z00C	SRA E 2 8 Z00C	SRA H 2 8 Z00C	SRA L 2 8 Z00C	SRA [HL] 2 16 Z00C	SRA A 2 8 Z00C
3x	SWAP B 2 8 Z000	SWAP C 2 8 Z000	SWAP D 2 8 Z000	SWAP E 2 8 Z000	SWAP H 2 8 Z000	SWAP L 2 8 Z000	SWAP [HL] 2 16 Z000	SWAP A 2 8 Z000	SRL B 2 8 Z00C	SRL C 2 8 Z00C	SRL D 2 8 Z00C	SRL E 2 8 Z00C	SRL H 2 8 Z00C	SRL L 2 8 Z00C	SRL [HL] 2 16 Z00C	SRL A 2 8 Z00C
4x	BIT 0, B 2 8 Z01-	BIT 0, C 2 8 Z01-	BIT 0, D 2 8 Z01-	BIT 0, E 2 8 Z01-	BIT 0, H 2 8 Z01-	BIT 0, L 2 8 Z01-	BIT 0, [HL] 2 12 Z01-	BIT 0, A 2 8 Z01-	BIT 1, B 2 8 Z01-	BIT 1, C 2 8 Z01-	BIT 1, D 2 8 Z01-	BIT 1, E 2 8 Z01-	BIT 1, H 2 8 Z01-	BIT 1, L 2 8 Z01-	BIT 1, [HL] 2 12 Z01-	BIT 1, A 2 8 Z01-
5x	BIT 2, B 2 8 Z01-	BIT 2, C 2 8 Z01-	BIT 2, D 2 8 Z01-	BIT 2, E 2 8 Z01-	BIT 2, H 2 8 Z01-	BIT 2, L 2 8 Z01-	BIT 2, [HL] 2 12 Z01-	BIT 2, A 2 8 Z01-	BIT 3, B 2 8 Z01-	BIT 3, C 2 8 Z01-	BIT 3, D 2 8 Z01-	BIT 3, E 2 8 Z01-	BIT 3, H 2 8 Z01-	BIT 3, L 2 8 Z01-	BIT 3, [HL] 2 12 Z01-	BIT 3, A 2 8 Z01-
6x	BIT 4, B 2 8 Z01-	BIT 4, C 2 8 Z01-	BIT 4, D 2 8 Z01-	BIT 4, E 2 8 Z01-	BIT 4, H 2 8 Z01-	BIT 4, L 2 8 Z01-	BIT 4, [HL] 2 12 Z01-	BIT 4, A 2 8 Z01-	BIT 5, B 2 8 Z01-	BIT 5, C 2 8 Z01-	BIT 5, D 2 8 Z01-	BIT 5, E 2 8 Z01-	BIT 5, H 2 8 Z01-	BIT 5, L 2 8 Z01-	BIT 5, [HL] 2 12 Z01-	BIT 5, A 2 8 Z01-
7x	BIT 6, B 2 8 Z01-	BIT 6, C 2 8 Z01-	BIT 6, D 2 8 Z01-	BIT 6, E 2 8 Z01-	BIT 6, H 2 8 Z01-	BIT 6, L 2 8 Z01-	BIT 6, [HL] 2 12 Z01-	BIT 6, A 2 8 Z01-	BIT 7, B 2 8 Z01-	BIT 7, C 2 8 Z01-	BIT 7, D 2 8 Z01-	BIT 7, E 2 8 Z01-	BIT 7, H 2 8 Z01-	BIT 7, L 2 8 Z01-	BIT 7, [HL] 2 12 Z01-	BIT 7, A 2 8 Z01-
8x	RES 0, B 2 8 ----	RES 0, C 2 8 ----	RES 0, D 2 8 ----	RES 0, E 2 8 ----	RES 0, H 2 8 ----	RES 0, L 2 8 ----	RES 0, [HL] 2 16 ----	RES 0, A 2 8 ----	RES 1, B 2 8 ----	RES 1, C 2 8 ----	RES 1, D 2 8 ----	RES 1, E 2 8 ----	RES 1, H 2 8 ----	RES 1, L 2 8 ----	RES 1, [HL] 2 16 ----	RES 1, A 2 8 ----
9x	RES 2, B 2 8 ----	RES 2, C 2 8 ----	RES 2, D 2 8 ----	RES 2, E 2 8 ----	RES 2, H 2 8 ----	RES 2, L 2 8 ----	RES 2, [HL] 2 16 ----	RES 2, A 2 8 ----	RES 3, B 2 8 ----	RES 3, C 2 8 ----	RES 3, D 2 8 ----	RES 3, E 2 8 ----	RES 3, H 2 8 ----	RES 3, L 2 8 ----	RES 3, [HL] 2 16 ----	RES 3, A 2 8 ----
Ax	RES 4, B 2 8 ----	RES 4, C 2 8 ----	RES 4, D 2 8 ----	RES 4, E 2 8 ----	RES 4, H 2 8 ----	RES 4, L 2 8 ----	RES 4, [HL] 2 16 ----	RES 4, A 2 8 ----	RES 5, B 2 8 ----	RES 5, C 2 8 ----	RES 5, D 2 8 ----	RES 5, E 2 8 ----	RES 5, H 2 8 ----	RES 5, L 2 8 ----	RES 5, [HL] 2 16 ----	RES 5, A 2 8 ----
Bx	RES 6, B 2 8 ----	RES 6, C 2 8 ----	RES 6, D 2 8 ----	RES 6, E 2 8 ----	RES 6, H 2 8 ----	RES 6, L 2 8 ----	RES 6, [HL] 2 16 ----	RES 6, A 2 8 ----	RES 7, B 2 8 ----	RES 7, C 2 8 ----	RES 7, D 2 8 ----	RES 7, E 2 8 ----	RES 7, H 2 8 ----	RES 7, L 2 8 ----	RES 7, [HL] 2 16 ----	RES 7, A 2 8 ----
Cx	SET 0, B 2 8 ----	SET 0, C 2 8 ----	SET 0, D 2 8 ----	SET 0, E 2 8 ----	SET 0, H 2 8 ----	SET 0, L 2 8 ----	SET 0, [HL] 2 16 ----	SET 0, A 2 8 ----	SET 1, B 2 8 ----	SET 1, C 2 8 ----	SET 1, D 2 8 ----	SET 1, E 2 8 ----	SET 1, H 2 8 ----	SET 1, L 2 8 ----	SET 1, [HL] 2 16 ----	SET 1, A 2 8 ----
Dx	SET 2, B 2 8 ----	SET 2, C 2 8 ----	SET 2, D 2 8 ----	SET 2, E 2 8 ----	SET 2, H 2 8 ----	SET 2, L 2 8 ----	SET 2, [HL] 2 16 ----	SET 2, A 2 8 ----	SET 3, B 2 8 ----	SET 3, C 2 8 ----	SET 3, D 2 8 ----	SET 3, E 2 8 ----	SET 3, H 2 8 ----	SET 3, L 2 8 ----	SET 3, [HL] 2 16 ----	SET 3, A 2 8 ----
Ex	SET 4, B 2 8 ----	SET 4, C 2 8 ----	SET 4, D 2 8 ----	SET 4, E 2 8 ----	SET 4, H 2 8 ----	SET 4, L 2 8 ----	SET 4, [HL] 2 16 ----	SET 4, A 2 8 ----	SET 5, B 2 8 ----	SET 5, C 2 8 ----	SET 5, D 2 8 ----	SET 5, E 2 8 ----	SET 5, H 2 8 ----	SET 5, L 2 8 ----	SET 5, [HL] 2 16 ----	SET 5, A 2 8 ----
Fx	SET 6, B 2 8 ----	SET 6, C 2 8 ----	SET 6, D 2 8 ----	SET 6, E 2 8 ----	SET 6, H 2 8 ----	SET 6, L 2 8 ----	SET 6, [HL] 2 16 ----	SET 6, A 2 8 ----	SET 7, B 2 8 ----	SET 7, C 2 8 ----	SET 7, D 2 8 ----	SET 7, E 2 8 ----	SET 7, H 2 8 ----	SET 7, L 2 8 ----	SET 7, [HL] 2 16 ----	SET 7, A 2 8 ----

Rysunek 2: Tablica rozmieszczenia instrukcji po opcodzie 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 półbajtów 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)
FEA0-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 rejestry 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. Rozwiązanie 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[5]

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 rejestry 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

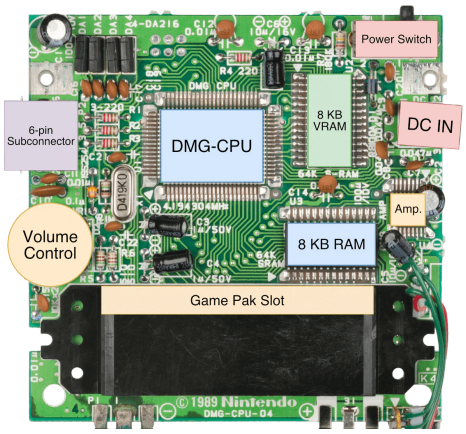
W przypadku jednoczesnego zgłoszenia wielu przerwań stosowany jest ustalony porządek priorytetów. Najwyższy priorytet posiada przerwanie V-Blank, a najniższy przerwanie kontrolera wejściowego. Taki schemat zapewnia deterministyczną obsługę zdarzeń związanych z generowaniem obrazu, które są krytyczne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

2.5.5 Rola systemu przerwań w architekturze konsoli

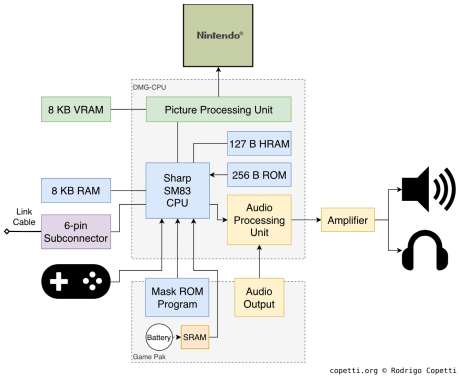
System przerwań stanowi podstawowy mechanizm synchronizacji komponentów sprzętowych konsoli Game Boy. Dzięki niemu możliwe jest efektywne współdziałanie procesora z kontrolerem graficznym, licznikami czasu oraz urządzeniami wejścia/wyjścia przy zachowaniu deterministycznego modelu wykonania programu. 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 kompomentów[1]



Rysunek 4: Schemat płyty głównej konsoli GameBoy[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 kartridża 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, kompoment 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 rozszerzający 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 wczesnym 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). Rozwiązanie 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 podtrzymania zasilania pamięci zapisu. Komunikacja z konsolą odbywa się za pośrednictwem złącza krawędziowego, które zapewnia dostęp do magistrali adresowej, magistrali danych oraz linii sterujących.

3 Model działania systemu Game Boy

3.1 Cykl pracy procesora

Procesor zastosowany w konsoli Game Boy jest 8-bitową jednostką opartą na zmodyfikowanej architekturze zbliżonej do Z80. Jego działanie jest determinowane przez cykl zegara systemowego oraz mechanizm wykonywania instrukcji w modelu sekwencyjnym. Zrozumienie cyklu pracy procesora jest kluczowe dla analizy działania całego systemu, ponieważ CPU koordynuje komunikację pomiędzy pamięcią, układem graficznym oraz urządzeniami wejścia/wyjścia.

3.1.1 Model cyklu rozkazowego

Podstawowy cykl pracy procesora można opisać jako powtarzającą się sekwencję trzech etapów:

1. pobranie instrukcji z pamięci,
2. dekodowanie instrukcji,
3. wykonanie instrukcji.

W pierwszym etapie procesor odczytuje kod operacji spod adresu wskazywanego przez licznik programu (PC). Następnie licznik programu zostaje inkrementowany, a pobrana wartość trafia do wewnętrznego rejestru instrukcji.

W drugim etapie następuje interpretacja kodu operacji oraz ustalenie wymaganych operandów. Proces dekodowania determinuje, czy instrukcja wymaga dodatkowych danych z pamięci oraz jakie jednostki funkcjonalne procesora zostaną zaangażowane w jej wykonanie.

Trzeci etap obejmuje właściwe wykonanie operacji arytmetycznej, logicznej lub sterującej. W zależności od typu instrukcji może to obejmować operacje na rejestrach, dostęp do pamięci lub zmianę przepływu sterowania programu.

3.1.2 Synchronizacja z zegarem systemowym

Procesor pracuje synchronicznie z zegarem systemowym, którego częstotliwość determinuje tempo wykonywania operacji. Każda instrukcja wymaga określonej liczby cykli zegara do wykonania. Czas realizacji instrukcji nie jest stały i zależy od jej złożoności oraz liczby operacji pamięciowych.

Istotnym elementem synchronizacji jest współdzielenie magistrali danych z innymi komponentami systemu. Dostęp do pamięci jest realizowany w ściśle określonych momentach cyklu zegara, co zapewnia spójność transmisji danych pomiędzy CPU a pozostałymi układami.

3.1.3 Dostęp do pamięci i operacje magistrali

Procesor komunikuje się z pamięcią poprzez magistralę adresową i magistralę danych. W trakcie cyklu rozkazowego mogą wystąpić operacje odczytu lub zapisu pamięci. Operacje te są wykonywane sekwencyjnie i blokują dostęp do magistrali dla innych komponentów na czas trwania transferu.

Mapowanie pamięci odgrywa istotną rolę w cyklu pracy procesora, ponieważ różne zakresy adresowe odpowiadają różnym typom zasobów systemowych, takim jak pamięć operacyjna, pamięć wideo oraz rejestry sterujące urządzeń peryferyjnych.

3.1.4 Obsługa przerwań

Cykl pracy procesora może zostać przerwany przez mechanizm przerwań sprzętowych. Wystąpienie przerwania powoduje czasowe wstrzymanie wykonywania bieżącego programu oraz zapis aktualnego stanu procesora na stosie.

Po zaakceptowaniu przerwania procesor przechodzi do procedury obsługi przerwania wskazanej w tablicy wektorów przerwań. Po zakończeniu obsługi następuje przywrócenie poprzedniego stanu procesora i kontynuacja wykonywania programu od miejsca przerwania.

3.1.5 Znaczenie cyklu pracy dla działania systemu

Deterministyczny charakter cyklu pracy procesora umożliwia precyzyjną synchronizację z układem graficznym oraz pozostałymi komponentami systemu. W szczególności liczba cykli zegara zużywana przez instrukcje wpływa bezpośrednio na generowanie obrazu oraz obsługę zdarzeń wejścia/wyjścia.

Analiza cyklu pracy procesora stanowi podstawę do modelowania działania systemu oraz jest niezbędna przy implementacji oprogramowania emulującego działanie konsoli.

3.2 Magistrala pamięciowa i charakterystyka działania pamięci

System pamięci konsoli Game Boy opiera się na wspólnej przestrzeni adresowej obsługiwanej przez procesor oraz na zestawie wyspecjalizowanych obszarów pamięci o odmiennych funkcjach. Komunikacja pomiędzy jednostką centralną a zasobami pamięciowymi realizowana jest za pomocą magistrali adresowej, magistrali danych oraz sygnałów sterujących, które determinują kierunek i tryb transferu informacji.

3.2.1 Organizacja magistrali pamięciowej

Magistrala adresowa umożliwia wskazanie konkretnej komórki pamięci w przestrzeni adresowej systemu. Procesor generuje adres, który identyfikuje zasób, z którego ma nastąpić odczyt danych lub do którego ma zostać wykonany zapis. Magistrala danych służy do przesyłania właściwych informacji pomiędzy procesorem a wskazanym komponentem systemu.

Operacje pamięciowe są synchronizowane z zegarem systemowym. W każdym cyklu dostępu do pamięci następuje ustalenie adresu, aktywacja sygnału sterującego oraz transfer danych. Ze względu na współdzielenie magistrali przez różne komponenty systemu dostęp do niej jest ściśle kontrolowany, co zapobiega konfliktom transmisji.

3.2.2 Mapowanie przestrzeni adresowej

Przestrzeń adresowa systemu jest podzielona na logiczne segmenty odpowiadające różnym typom zasobów sprzętowych. Poszczególne zakresy adresów są przypisane do pamięci programu, pamięci operacyjnej, pamięci wideo oraz rejestrów sterujących urządzeń peryferyjnych.

Mapowanie pamięci pełni funkcję mechanizmu integrującego różnorodne komponenty sprzętowe w jednolitą przestrzeń adresową. Dzięki temu procesor może komunikować się z układami systemowymi za pomocą standardowych operacji odczytu i zapisu, bez konieczności stosowania odrębnych instrukcji wejścia/wyjścia.

3.2.3 Rodzaje pamięci w systemie

W architekturze systemu można wyróżnić kilka podstawowych typów pamięci:

- pamięć programu (ROM), zawierająca kod wykonywalny oraz dane statyczne,
- pamięć operacyjna (RAM), wykorzystywana do przechowywania danych tymczasowych,
- pamięć wideo (VRAM), przeznaczona do przechowywania danych graficznych,
- rejestry urządzeń peryferyjnych odwzorowane w przestrzeni adresowej.

Każdy z wymienionych typów pamięci charakteryzuje się odmiennym przeznaczeniem funkcjonalnym oraz sposobem dostępu. W szczególności pamięć wideo jest współdzielona z układem graficznym, co wprowadza ograniczenia czasowe w dostępie do niej przez procesor.

3.2.4 Charakterystyka operacji pamięciowych

Operacje pamięciowe wykonywane przez procesor obejmują odczyt danych, zapis danych oraz transfery blokowe realizowane sekwencyjnie. Czas trwania operacji zależy od rodzaju pamięci oraz liczby cykli zegara wymaganych do realizacji transferu.

Dostęp do pamięci może być ograniczony przez stan systemu, w szczególności przez aktywność układów współdzielących magistralę. W określonych fazach pracy systemu wybrane obszary pamięci mogą być czasowo niedostępne dla procesora lub dostępne jedynie w określonych warunkach synchronizacji.

3.2.5 Rola systemu pamięci w działaniu konsoli

Architektura pamięci stanowi podstawę wymiany informacji pomiędzy wszystkimi komponentami systemu. Determinuje ona sposób wykonywania programu, obsługi grafiki oraz komunikacji z urządzeniami wejścia/wyjścia.

Jednolita przestrzeń adresowa oraz deterministyczny model dostępu do pamięci umożliwiają przewidywalne działanie systemu w czasie rzeczywistym. Właściwości te mają istotne znaczenie zarówno dla analizy architektury sprzętowej, jak i dla implementacji oprogramowania odwzorowującego działanie systemu.

3.3 Generowanie klatki obrazu

Generowanie obrazu w konsoli Game Boy jest realizowane przez układ graficzny (PPU), który współpracuje z pamięcią wideo oraz pamięcią atrybutów obiektów. Proces ten przebiega cyklicznie i jest ściśle zsynchronizowany z zegarem systemowym oraz działaniem procesora. Obraz powstaje poprzez sekwencyjne renderowanie tła, a następnie nakładanie obiektów ruchomych zapisanych w pamięci OAM.

3.3.1 Format zapisu grafiki w pamięci VRAM

Podstawową jednostką reprezentacji obrazu w systemie jest tzw. kafel (ang. tile). Każdy kafel reprezentuje fragment obrazu o wymiarach 8×8 pikseli. Dane kafli przechowywane są w pamięci wideo (VRAM) w postaci uporządkowanej tablicy bajtów.

Każdy wiersz kafla kodowany jest przy użyciu dwóch bajtów, które reprezentują dwubitową informację o kolorze każdego piksela w wierszu. Pierwszy bajt zawiera najmniej znaczące bity wartości koloru, natomiast drugi bajt zawiera bity najbardziej znaczące. Odczyt odpowiadających sobie bitów z obu bajtów pozwala określić indeks koloru dla pojedynczego piksela.

Układ graficzny interpretuje dane kafli na podstawie mapy tła, która przechowuje indeksy kafli oraz ich rozmieszczenie na ekranie. Dzięki temu możliwe jest wielokrotne wykorzystanie tych samych danych graficznych w różnych miejscach obrazu.

3.3.2 Renderowanie tła

Proces generowania klatki rozpoczyna się od renderowania tła. Układ graficzny przetwarza mapę tła linia po linii, odczytując odpowiednie indeksy kafli oraz ich dane z pamięci VRAM. Dla każdej linii obrazu wyznaczany jest odpowiedni fragment kafla, który następnie przekształcany jest w wartości kolorów pikseli.

Renderowanie odbywa się w sposób deterministyczny i jest zsynchronizowane z aktualną pozycją skanowania ekranu. W wyniku tego procesu powstaje podstawowa warstwa obrazu, która stanowi bazę dla dalszego etapu renderowania.

3.3.3 Pamięć OAM i reprezentacja obiektów

OAM (Object Attribute Memory) jest wyspecjalizowanym obszarem pamięci przeznaczonym do przechowywania informacji o obiektach ruchomych, określanych jako sprite'y. Pamięć ta zawiera zestaw rekordów opisujących właściwości graficzne oraz położenie obiektów wyświetlanych na ekranie.

Każdy wpis w pamięci OAM zawiera:

- współrzędną pionową obiektu na ekranie,
- współrzędną poziomą obiektu,
- indeks kafla opisującego grafikę obiektu,
- zestaw atrybutów określających sposób renderowania.

Atrybuty obiektu mogą obejmować informacje o priorytecie wyświetlania, odbiciu lustrzanym oraz wyborze palety kolorów. Układ graficzny odczytuje zawartość pamięci OAM podczas procesu generowania obrazu i wykorzystuje ją do nałożenia obiektów na wcześniej wyrenderowane tło.

3.3.4 Renderowanie obiektów

Po wygenerowaniu warstwy tła układ graficzny przystępuje do renderowania obiektów zapisanych w pamięci OAM. Dla każdej linii obrazu wybierane są obiekty znajdujące się w jej obszarze, a następnie ich dane graficzne są pobierane z pamięci VRAM.

Renderowanie obiektów polega na nałożeniu ich pikseli na istniejący obraz tła zgodnie z regułami priorytetu. W przypadku kolizji pikseli o różnych źródłach decyzja o widoczności zależy od atrybutów obiektu oraz wartości koloru tła.

3.3.5 DMA i bezpośredni transfer danych do OAM

W celu usprawnienia transferu danych do pamięci OAM system wykorzystuje mechanizm DMA (Direct Memory Access). Mechanizm ten umożliwia bezpośrednie kopiowanie bloków danych z określonego obszaru pamięci do pamięci OAM bez konieczności wykonywania instrukcji kopiowania przez procesor.

Transfer DMA polega na automatycznym odczycie kolejnych bajtów ze źródłowego obszaru pamięci oraz ich zapisie do kolejnych komórek pamięci OAM. W trakcie operacji DMA procesor ma ograniczony dostęp do magistrali pamięci, co zapewnia spójność przesyłanych danych.

Zastosowanie mechanizmu bezpośredniego transferu bajtów umożliwia szybkie aktualizowanie danych obiektów, co jest szczególnie istotne w systemach czasu rzeczywistego, gdzie konieczna jest synchronizacja zmian graficznych z generowaniem kolejnych klatek obrazu.

3.3.6 Znaczenie procesu generowania obrazu

Proces generowania klatki obrazu stanowi wynik ścisłej współpracy pamięci wideo, pamięci OAM oraz układu graficznego. Deterministyczny charakter renderowania oraz uporządkowany dostęp do danych umożliwiają przewidywalne odwzorowanie stanu systemu graficznego w każdej chwili pracy konsoli.

Zrozumienie sposobu generowania obrazu ma istotne znaczenie dla analizy architektury systemu oraz implementacji oprogramowania emulującego działanie układu graficznego.

3.4 System przerwań

Mechanizm przerwań w konsoli Game Boy umożliwia asynchroniczną reakcję procesora na zdarzenia sprzętowe zachodzące w systemie. Zamiast ciągłego odpytywania stanu urządzeń peryferyjnych, procesor może wykonywać program główny, a w momencie wystąpienia określonego zdarzenia przejść do dedykowanej procedury obsługi przerwania. Rozwiązanie to zwiększa efektywność wykorzystania czasu procesora oraz zapewnia deterministyczną obsługę zdarzeń czasowych.

3.4.1 Warunki występowania przerwań

Przerwanie może zostać wygenerowane przez układ sprzętowy po spełnieniu określonego warunku logicznego, takiego jak zakończenie operacji wejścia/wyjścia, osiągnięcie zadanego stanu licznika czasowego lub zmiana stanu kontrolera wejściowego. Aby przerwanie zostało obsłużone, muszą zostać spełnione trzy warunki:

1. źródło przerwania zgłosi żądanie,
2. odpowiedni typ przerwania jest odblokowany w rejestrze maskującym,
3. globalny mechanizm obsługi przerwań w procesorze jest aktywny.

Jeżeli wszystkie warunki są spełnione, procesor kończy wykonywanie bieżącej instrukcji, zapisuje aktualny stan wykonania programu oraz przechodzi do procedury obsługi przerwania.

3.4.2 Rodzaje przerwań

System przewiduje zestaw sprzętowych źródeł przerwań odpowiadających kluczowym funkcjom konsoli:

- przerwanie synchronizacji pionowej obrazu (V-Blank),
- przerwanie związane ze stanem układu graficznego (LCD STAT),
- przerwanie licznika czasowego (Timer),
- przerwanie zakończenia transmisji szeregowej,
- przerwanie kontrolera wejścia użytkownika (Joypad).

Każdy typ przerwania posiada przypisany wektor, czyli stały adres w pamięci programu, pod którym rozpoczyna się procedura jego obsługi. Priorytet przerwań jest ustalony sprzętowo i determinuje kolejność ich obsługi w przypadku jednoczesnego wystąpienia kilku zdarzeń.

3.4.3 Przebieg obsługi przerwania

Obsługa przerwania przebiega według ściśle określonej sekwencji działań wykonywanych przez procesor:

1. zakończenie wykonywania bieżącej instrukcji,
2. zapis aktualnej wartości licznika programu na stosie,
3. czasowe zablokowanie dalszych przerwań,
4. przejście do adresu procedury obsługi przerwania,
5. wykonanie kodu obsługi zdarzenia,
6. przywrócenie poprzedniego stanu programu i kontynuacja wykonania.

Zapis stanu programu na stosie umożliwia powrót do przerwanego miejsca po zakończeniu obsługi. Instrukcja powrotu z przerwania przywraca licznik programu oraz ponownie aktywuje globalny mechanizm przerwań.

3.4.4 Funkcje realizowane przez przerwania

Przerwania pełnią w systemie rolę mechanizmu synchronizującego działanie procesora z procesami zachodzącymi w układach peryferyjnych. Do najważniejszych funkcji realizowanych za ich pomocą należą:

- synchronizacja zmian graficznych z cyklem generowania obrazu,
- odmierzanie czasu oraz generowanie zdarzeń okresowych,
- obsługa transmisji danych pomiędzy komponentami systemu,
- reagowanie na działania użytkownika.

W szczególności przerwanie synchronizacji pionowej umożliwia bezpieczną aktualizację danych graficznych w pamięci wideo w momentach, gdy układ graficzny nie odczytuje danych do generowania obrazu. Mechanizm przerwań zapewnia tym samym spójność danych oraz stabilność działania systemu w czasie rzeczywistym.

3.4.5 Znaczenie mechanizmu przerwań

Zastosowanie przerwań pozwala na efektywne zarządzanie współbieżnymi zdarzeniami w systemie o ograniczonych zasobach sprzętowych. Deterministyczna obsługa zdarzeń oraz priorytetyzacja źródeł przerwań stanowią podstawę prawidłowego działania konsoli oraz umożliwiają implementację oprogramowania wymagającego precyzyjnej synchronizacji czasowej.

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 (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).