1

Wprowadzenie

1.1. Wstęp

We współczesnej technice powszechne jest wykorzystanie programowalnych układów cyfrowych. Często stanowią one jeden z elementów większego urządzenia bądź systemu i dlatego noszą nazwę „systemów wbudowanych". Zdarza się, że obecność takiego układu jest wręcz niezauważalna dla użytkownika i nie zdaje on sobie nawet sprawy, że gdzieś we wnętrzu danego urządzenia kryje się niewielki system komputerowy. Systemy wbudowane mogą być realizowane na różne sposoby, jednakże zdecydowana większość wykorzystuje mikrokontrolery, czyli cyfrowe systemy mikroprocesorowe zrealizowane w postaci pojedynczego układu scalonego. Zaskoczeniem może być fakt, że są one obecnie najbardziej rozpowszechnioną formą systemów komputerowych. Najlepiej obrazują to dane dotyczące liczby sprzedanych sztuk różnych procesorów. Okazuje się, że rocznie sprzedaje się 20 razy więcej mikrokontrolerów niż procesorów przeznaczonych do „zwykłych" komputerów.

Pierwsze mikrokontrolery pojawiły się na rynku w połowie lat 70. XX w. Aż do końca lat 90. dominującą pozycję miały układy 8-bitowe, a układy 32-bitowe, przede wszystkim ze względu na wysokie ceny, były stosowane rzadko, tylko w najbardziej wymagających systemach. Ostatnie lata przyniosły jednak istotne zmiany. Sprzedaż układów 32-bitowych zaczęła dynamicznie rosnąć i obecnie stanowią one blisko 50% ogólnej liczby sprzedawanych mikrokontrolerów. Dzieje się tak głównie za sprawą wprowadzania przez producentów sprzętu elektronicznego coraz bardziej zaawansowanych urządzeń, takich jak telefony komórkowe, systemy nawigacji satelitarnej, systemy audio i video, konsole do gier itp., które wymagają układów sterujących o dużej wydajności.

Znaczne zmiany zaszły także pod względem stosowanych architektur mikrokontrolerów. Dawniej każdy liczący się producent układów elektronicznych oferował procesory wykorzystujące własne konstrukcje rdzenia. Obecnie w ofercie większości wytwórców znaleźć można układy wyposażone w 32-bitowe rdzenie zaprojektowane przez firmę ARM. Niektórzy producenci w ogóle nie produkują procesorów 32-bitowych innych, niż z rdzeniem ARM. Rdzenie tej rodziny zaczęły zyskiwać dużą popularność od połowy lat 90. XX w., a ich najbardziej znaną wersją był ARM7TDMI. Od kilku lat procesory z tym rdzeniem są jednak sukcesywnie wypierane z rynku przez nowszą rodzinę rdzeni – ARM Cortex.

Rosnąca popularność rodziny ARM i jej udział w rynku powoduje, że być może wkrótce znajomość tej architektury będzie niemal obowiązkowa wśród konstruktorów i programistów tworzących systemy wbudowane. Potwierdzeniem i zapowiedzią przyszłych trendów może być na przykład fakt, że już obecnie procesory z rdzeniami ARM mają ponad 10% udziału w globalnym rynku mikrokontrolerów, ponad 90% udziału w rynku mikrokontrolerów 32-bitowych i ponad 98% udziału w rynku procesorów przeznaczonych dla telefonów komórkowych. Warto więc poświęcić nieco czasu na zaznajomienie się z jedną z rodzin mikrokontrolerów wykorzystujących rdzeń ARM Cortex-M3, jaką stanowią układy STM32 produkowane przez firmę STMicroelectronics.

Niniejsza książka przeznaczona jest przede wszystkim dla osób początkujących, które nie miały jeszcze styczności z mikrokontrolerami rodziny ARM a zwłaszcza

z układami z rdzeniem Cortex-M3. Zawiera ona ponad 30 rozbudowanych ćwiczeń, które pozwalają poznać możliwości układów serii STM32F1xx. Ćwiczenia przygotowano w języku C, przy czym zakłada się, że czytelnik miał już z nim styczność i zna podstawowe pojęcia związane z programowaniem w tym języku.

Oprócz ćwiczeń, w książce zawarto także przydatne do ich wykonania opisy. Dotyczą one zastosowanych podukładów peryferyjnych takich jak m.in. porty wejścia/wyjścia, układy licznikowe, przetworniki A/C, interfejsy komunikacyjne oraz mechanizmów takich jak przerwania czy DMA. Ponadto, opisano i pokazano także sposób wykorzystania kilku ciekawych, zewnętrznych modułów rozszerzających (m.in. klawiatura, alfanumeryczne i graficzne wyświetlacze LCD, karty SD, czujnik położenia, akcelerometr, barometr, układ Bluetooth). Ta część niniejszej pozycji może być interesująca także dla osób o nieco wyższym stopniu zaawansowania w programowaniu mikrokontrolerów.

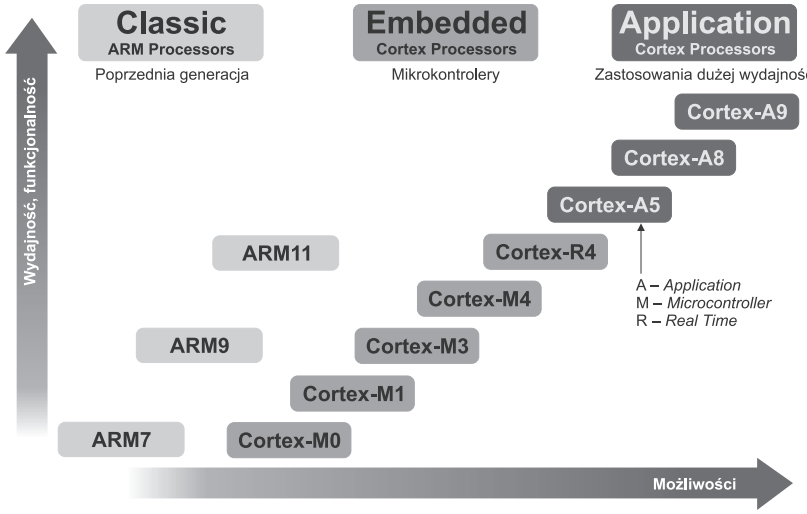
Poszczególne ćwiczenia są pomyślane w ten sposób, że nakreślony jest zarys rozwiązania, przedstawione są kluczowe fragmenty kodu oraz elementy nowo wprowadzane, w stosunku do poprzednich ćwiczeń. Pozostała część większości ćwiczeń wymaga pewnego nakładu samodzielnej pracy, ułatwionej jednak licznymi podpowiedziami i wyjaśnieniami. Wykonanie wszystkich zadań z książki powinno zająć około 60…70 godzin. Sposób prowadzenia ćwiczeń pozwala zarówno na samodzielną naukę programowania jak i na wykorzystanie książki np. jako podstawy do prowadzenia kursów i zajęć laboratoryjnych w szkołach bądź na uczelniach.

1.2. Mikrokontrolery rodziny ARM

Rdzenie rodziny ARM są projektowane w firmie ARM Ltd. Jest to najbardziej znany i popularny wyrób tej firmy. Istotnym jest fakt, iż firma ta nie zajmuje się produkcją, a jedynie opracowuje układy cyfrowe, w szczególności w postaci rozwiązań typu *IPcore* (*Intellectual Property Core*). Procesory wykorzystujące opracowane przez ARM rdzenie są natomiast produkowane przez większość najważniejszych producentów układów elektronicznych na świecie (m.in. Freescale, NXP, STMicroelectronics, Texas Instruments, Toshiba, Zilog). Warto zauważyć, że na końcowy produkt, czyli gotowy mikrokontroler, składa się nie tylko rdzeń, ale także wiele dodatkowych podukładów peryferyjnych, które dodawane są już przez poszczególnych producentów. Stąd bierze się bardzo duża różnorodność oferty mikrokontrolerów, mimo że podstawowym ich elementem są takie same rdzenie ARM.

Układy z rdzeniem ARM są 32-bitowymi procesorami RISC. Cechują się prostą konstrukcją, w której wykorzystywana jest mniejsza (w porównaniu z rozwiązaniami konkurencyjnymi) liczba tranzystorów, co pozwala uzyskać niski pobór mocy i mały rozmiar procesora przy jednoczesnej dużej jego wydajności. Poszczególne wersje rdzeni różnią się przede wszystkim zbiorem obsługiwanych instrukcji i zastosowanymi technologiami, a co za tym idzie – możliwościami i wydajnością. Wyróżnić można wśród nich:

– podstawowy zestaw instrukcji 32-bitowych ARM,
– instrukcje *Thumb* i *Thumb2* – specjalne zestawy instrukcji 16-bitowych,
– instrukcje NEON – instrukcje typu SIMD,



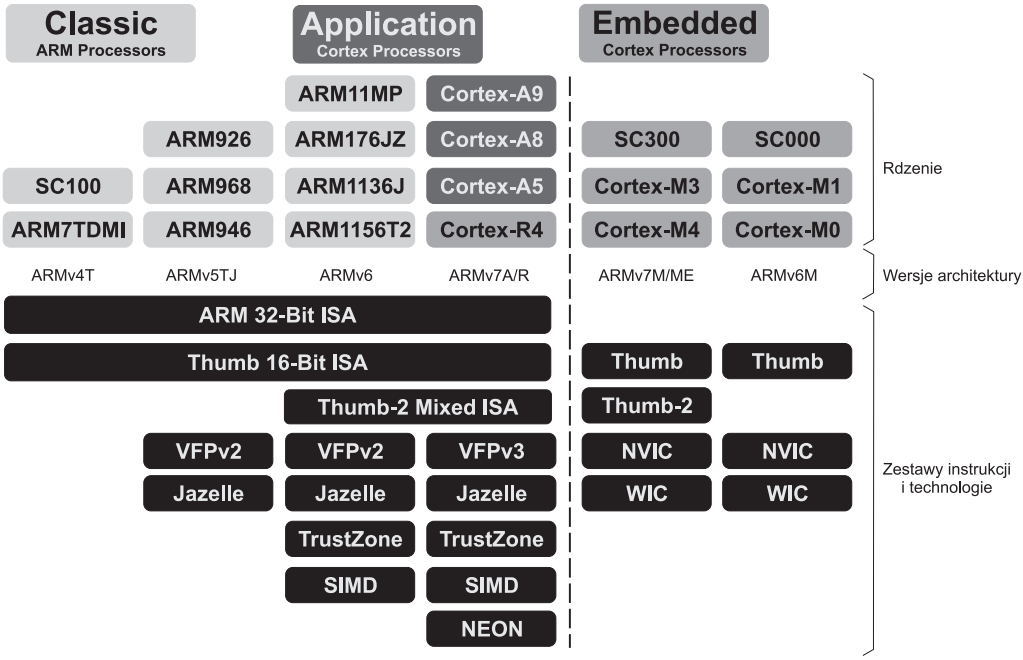
Rys. 1.1. Aktualnie oferowane rdzenie ARM

– instrukcje VFP – specjalne instrukcje do operacji zmiennoprzecinkowych,
– technologię *Jazelle* – sprzętową obsługę kodu bajtowego utworzonego w języku Java,
– instrukcje wspomagające przetwarzanie sygnałów (DSP),
– technologię *TrustZone* – wsparcie dla aplikacji wymagających wysokiego bezpieczeństwa (wykorzystywane np. w systemach bankowych),
– NVIC – wielopoziomowy kontroler przerwań.

Szukając informacji o procesorach ARM można napotkać pewną trudność w rozróżnieniu stosowanych oznaczeń. Wynika to stąd, że producenci procesorów najczęściej operują oznaczeniami wersji rdzenia (np. ARM7TDMI, Cortex-M3 itp.), natomiast firma ARM preferuje oznaczenia wersji architektury rdzenia (np. ARMv4T, ARMv7M itp.). Podobieństwo tych nazw może wprowadzić nieco zamieszania. Na **rysunku 1.1** przedstawiono zestawienie aktualnie oferowanych przez ARM wersji rdzenia, a na **rysunku 1.2** – wersje rdzeni wraz z zastosowanymi w nich architekturami i obsługiwanymi zestawami instrukcji.

Obecnie rozwijana rodzina Cortex składa się z trzech podrodzin:

– **Cortex-Ax** – (*Application*) – rdzenie najbardziej zaawansowane, o najwyższej wydajności, przeznaczone do zastosowań z systemami operacyjnymi (np. Symbian, Linux, Windows Embedded); dodatkowo procesory te mają sprzętowe wsparcie dla programów napisanych w języku Java, co predestynuje je zwłaszcza do stosowania w telefonach komórkowych,
– **Cortex-Mx** – (*Microcontroller*) – rdzenie zaprojektowane z myślą o minimalizacji ceny przy zachowaniu dużej wydajności, przeznaczone zwłaszcza do zastosowań przemysłowych; cechuje je przede wszystkim brak obsługi głównego zestawu 32-bitowych instrukcji ARM,



Rys. 1.2. Wersje rdzeni ARM, ich architektura i obsługiwane zestawy instrukcji

– **Cortex-Rx** – (*Real Time*) – rdzenie dedykowane dla systemów czasu rzeczywistego, w których krytycznym parametrem jest czas reakcji systemu, przeznaczone w szczególności do zastosowania w motoryzacji.

Liczba w miejscu litery *x* oznacza wersję rdzenia.

1.3. Architektura rdzenia ARM Cortex-M3

1.3.1. Najważniejsze cechy architektury Cortex-M3

Rdzeń Cortex-M3 wykorzystuje architekturę ARMv7M. Pod względem organizacji pamięci jest to architektura harwardzka, tzn. pamięć zawierająca kod programu (Flash) i pamięć danych (SRAM) są rozdzielone i dostęp do nich odbywa się poprzez osobne magistrale. Niezależność obszarów pamięci daje m.in. możliwość równoległego wykonywania operacji na obu blokach pamięci, co pozwala zwiększyć wydajność systemu. W przypadku procesorów ARM istnieje jednak możliwość umieszczenia kodu programu w obszarze SRAM. Wadą takiego rozwiązania jest to, że program może działać nieco wolniej, gdyż procesor jest zoptymalizowany w taki sposób, by pobierał kod z pamięci Flash, a nie SRAM. Ponadto pamięci SRAM jest kilkukrotnie mniej niż pamięci Flash, więc można w niej zmieścić tylko stosunkowo nieduże programy. Należy też pamiętać, że po wyłączeniu zasilania, zawartość SRAM jest tracona. Podstawową zaletą opisywanego rozwiązana jest natomiast wydłużenie czasu życia pamięci Flash. Pamięć ta ma ograniczoną żywotność, która zwykle wynosi kilkanaście do kilkudziesięciu tysięcy cykli zapisu. Na etapie

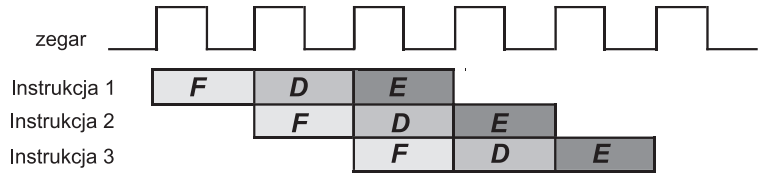
testowania programu, gdy kod jest wielokrotnie zmieniany, można więc zaoszczędzić dość znaczną liczbę tych cykli.

Jak już wspomniano, procesory ARM są procesorami RISC. Architekturę tę nazywa się niekiedy architekturą *Load/Store*. Procesor nie wykonuje operacji wprost na komórkach pamięci i nie ma też możliwości przesłania danych bezpośrednio pomiędzy komórkami pamięci. Dane muszą zostać najpierw załadowane (*load*) do rejestrów procesora, a po operacji odesłane (*store*) do pamięci – stąd nazwa architektury. Pozornie może się wydawać, że generuje to więcej operacji, ale dzięki temu lista instrukcji nie musi zawierać wielu złożonych poleceń. To z kolei pozwala na szybsze dekodowanie instrukcji i ich wykonanie.

Lista instrukcji procesorów Cortex-M3 zawiera około 130 instrukcji z zestawów *Thumb* i *Thumb2*. Instrukcje te operują zarówno na danych 16-, jak i 32-bitowych, ale same kody instrukcji są 16-bitowe. Można dzięki temu zaoszczędzić pamięć, ponieważ kod programu jest w niej lepiej „upakowany". W skrajnych przypadkach można uzyskać nawet 40% mniejszą zajętość pamięci niż dla takiego samego kodu, ale zapisanego z użyciem instrukcji 32-bitowych. Dodatkowo, krótsze kody instrukcji przyczyniają się do ogólnej poprawy wydajności procesora. Warto zauważyć, że rdzenie Cortex-M nie obsługują „normalnej", podstawowej listy instrukcji 32--bitowych ARM ani jej rozszerzeń (np. NEON czy *Jazelle*). Natomiast np. rdzenie Cortex-A pozwalają „mieszać" instrukcje 32- i 16-bitowe w ramach jednego programu, co daje możliwość optymalizacji rozmiaru kodu przy zachowaniu pełnych możliwości procesora.

Kolejną cechą wyróżniającą mikrokontrolery z rdzeniem Cortex-M, która zwiększa ich wydajność, jest wykorzystanie mechanizmu przetwarzania potokowego. Polega on na tym, że każda instrukcja jest dzielona na 3 etapy: pobranie instrukcji (*Fetch*), jej zdekodowanie (*Decode*) i wykonanie (*Execute*). Gdy jedna instrukcja jest w fazie wykonania, następna może być już dekodowana, a jeszcze następna – pobierana. Przedstawiono to obrazowo na **rysunku 1.3**. Dodatkowo, procesor potrafi wykonywać instrukcje w sposób spekulatywny, tzn. zawiera on wbudowany układ przewidywania rozgałęzień i skoków w programie, dzięki któremu może wykonać „na zapas" instrukcje znajdujące się za rozgałęzieniem.

Jeszcze jedną przydatną i zwiększającą wydajność cechą rdzeni Cortex jest sprzętowe wykonywanie operacji dzielenia. Operacja ta jest stosunkowo złożona i dlatego wiele prostszych i tańszych układów mikroprocesorowych nie ma jej zaimplementowanej. Wykonanie dzielenia w takiej sytuacji wymaga napisania dość rozbudowanego kodu programu. Umieszczenie w procesorach ARM Cortex jednostki dzielącej zdecydowanie upraszcza wykonanie tej operacji. Należy jednak pamiętać, że jest to



Rys. 1.3. Przetwarzanie potokowe

tylko dzielnie liczb stałoprzecinkowych, ponieważ procesor nie obsługuje zmiennoprzecinkowych typów danych.

1.3.2. Rejestry i organizacja pamięci

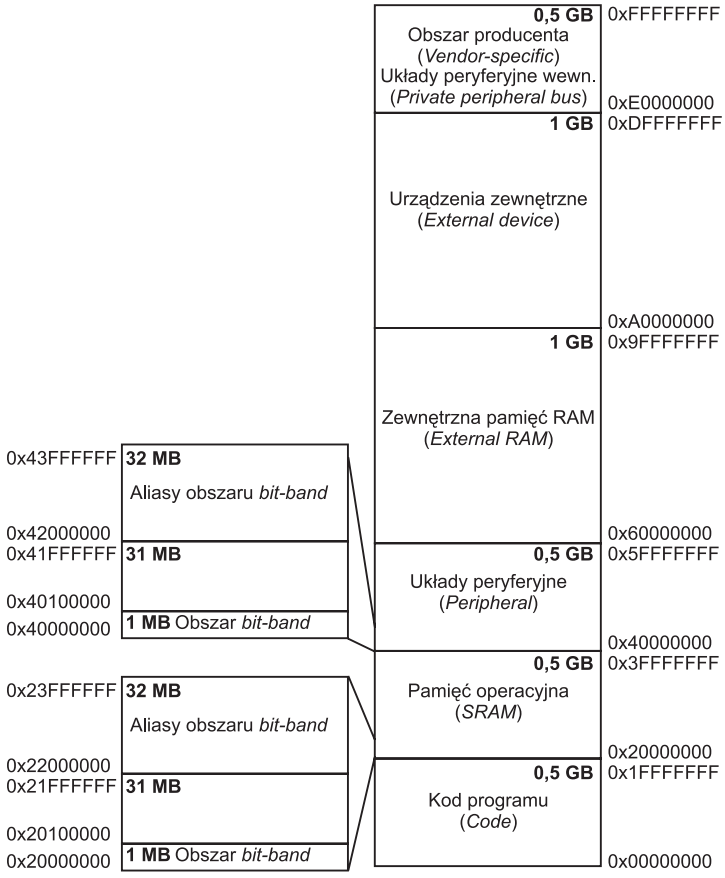
Podczas wykonywania obliczeń procesor operuje na danych umieszczonych w rejestrach. Rdzeń Cortex-M3 jest wyposażony w szesnaście 32-bitowych rejestrów podstawowych oznaczonych od R0 do R15. Rejestry od R0 do R12 są rejestrami ogólnego przeznaczenia (*General Purpose Register*). Spośród nich rejestry R0...R7 dostępne są dla wszystkich instrukcji, pozostałe zaś, tylko dla niektórych. Kolejny rejestr, R13, jest tzw. wskaźnikiem stosu. W rzeczywistości są to dwa rejestry, z których w danym momencie widoczny jest tylko jeden, tj. MSP lub PSP. MSP (*Main Stack Pointer*) jest rejestrem domyślnym, używanym przez przerwania i jądro systemu operacyjnego. Z kolei PSP (*Process Stack Pointer*) jest używany przez program użytkownika, jeśli na mikrokontrolerze działa system operacyjny, a program użytkownika jest uruchamiany za jego pośrednictwem. Dzięki zastosowaniu dwóch rejestrów stosu możliwe jest tworzenie bezpieczniejszych systemów, ponieważ program użytkownika nie ma dostępu do stosu systemu operacyjnego, a więc nie może go uszkodzić w przypadku wystąpienia jakiegoś błędu. Rejestr R14 (*Link Register*) zawiera tzw. adres powrotu. Dzięki niemu, procesor wie, w które miejsce wykonywanego kodu programu należy powrócić po zakończeniu wykonywania podprocedury lub procedury obsługi przerwania. Ostatni z rejestrów podstawowych, R15, to licznik rozkazów (*Program Counter*) i zawiera adres aktualnie wykonywanej instrukcji.

Oprócz rejestrów podstawowych, rdzeń zawiera wiele rejestrów specjalnych, które służą do sterowania wykonaniem programu oraz pracą procesora. Wśród nich można np. wyróżnić grupę rejestrów xPSR (*Program Status Register*), odpowiedzialnych m.in. za przechowywanie wyniku operacji porównania i przeniesienia czy wykorzystywanych do obsługi instrukcji warunkowych i przerwań. Inną grupę rejestrów stanowią rejestry związane ze śledzeniem wykonywania programu, także w trybie pracy krokowej, które wykorzystywane są podczas debugowania.

Wszystkie rdzenie ARM Cortex mają tak samo zorganizowaną przestrzeń adresową. Łącznie może ona obejmować 4 GB pamięci, jest jednak podzielona na segmenty o z góry zdefiniowanym przeznaczeniu. Najważniejsze obszary to:

– *Code* – obszar kodu programu (pamięć *Flash*),
– *SRAM* – pamięć operacyjna,
– *Peripheral* – porty i urządzenia peryferyjne,
– *External RAM* – pamięć zewnętrzna,
– *External device* – urządzenia zewnętrzne.

Podział i zakres adresów obejmujący poszczególne obszary pamięci (tzw. mapę pamięci) pokazano na **rysunku 1.4**. Możliwe jest inne skonfigurowanie przeznaczenia poszczególnych obszarów (np. umieszczenie danych w przestrzeni adresowej przeznaczonej dla kodu programu), ale są to rozwiązania zmniejszające efektywność działania procesora i w związku z tym stosowane przede wszystkim w przypadkach nietypowych.



Rys. 1.4. Mapa pamięci rdzenia Cortex-M3

Podczas przeglądania mapy pamięci, uwagę zwracają obszary *bit-band* i ich aliasy. Obszary te nazywane są także obszarami o dostępie atomowym lub bitowym. Zazwyczaj pamięć w systemach cyfrowych jest podzielona na komórki o pojemności 8 bitów. Każda z komórek ma określony adres, dzięki któremu można jednoznacznie określić, do której z nich chcemy się odwołać. Taka organizacja nie pozwala jednak uzyskać dostępu bezpośrednio do pojedynczych bitów słowa danych. By móc zmodyfikować wartość pojedynczego bitu, należy odczytać zawartość całej komórki pamięci, zmienić wartość interesującego nas bitu i ponownie zapisać w pamięci całe słowo. Rozwiązaniem ułatwiającym operowanie na pojedynczych bitach są obszary *bit-band*, w których każdy bit ma swój własny adres. Jest to szczególnie ważne i przydatne w systemach zbudowanych z wykorzystaniem mikrokontrolerów, gdyż tego typu operacje są tam bardzo częste. Oprócz dostępu bitowego, do obszarów *bit-band* można się oczywiście odwołać także według normalnych zasad adresowania.

Wyjaśnienia wymaga jeszcze sposób adresowania bitów w obszarze o dostępie atomowym. W tym celu stosuje się następujący wzór:

$$adres\_bitu = poczatek\_obszaru\_bitband + przesuniecie\_bajtu \times 32 + numer\_bitu \times 4,$$

gdzie:

adres_bitu	–	adres, pod jakim należy zapisywać i spod którego można odczytywać bit,
poczatek_obszaru_bitband	–	0x20000000 dla obszaru *bit-band* w pamięci SRAM oraz 0x40000000 dla obszaru *bit-band* w segmencie urządzeń peryferyjnych,
przesuniecie_bajtu	–	przesuniecie (*offset*) w stosunku do początku regionu *bit-band*,
numer_bitu	–	pozycja bitu w słowie, do którego chcemy uzyskać dostęp.

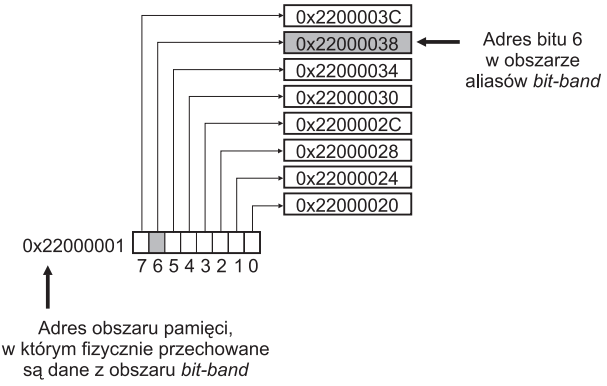
Na przykład, jeśli chcemy ustawić bit 6 komórki o adresie 0x2000001 (*offset* = 1), to zgodnie z przedstawionym wzorem, powinniśmy dokonać zapisu nowej wartości bitu pod adresem 0x22000038:

adres_bitu = 0x22000000 + 0x01 × 0x20 + 0x06 × 0x04 = 0x22000038.

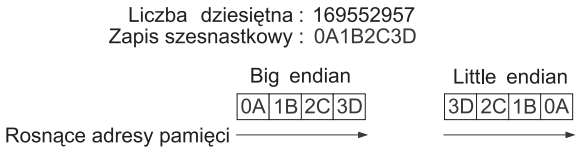
W zrozumieniu powyższych zależności pomocny może być także **rysunek 1.5**.

Przeglądając mapę pamięci warto zwrócić uwagę, że zawiera one nie tylko adresy typowych obszarów pamięci, ale również obszar adresów przeznaczony dla urządzeń peryferyjnych. W obszarze tym znajdują się rejestry danych i rejestry sterujące poszczególnych układów peryferyjnych mikrokontrolera. Na przykład w zakresie adresów od 0x40010800 do 0x40010BFF znajdują się rejestry portu GPIOA, a w obszarze od 0x40005400 do 0x400057FF – rejestry układu pierwszego kontrolera interfejsu I^2C. Dzięki takiej organizacji pamięci, dostęp do tych rejestrów jest, z punktu widzenia programisty, taki sam jak dostęp do dowolnej komórki pamięci. Jest to więc rozwiązanie bardzo wygodne.

Mikrokontrolery ARM są układami 32-bitowymi. Oznacza to, że podstawową długością słowa, na jakiej operują, jest słowo 4-bajtowe. Potrafią one, oczywiście, operować również na słowach 1- i 2-bajtowych. Ponieważ pamięć jest podzielona na bloki 1-bajtowe, w przypadku, gdy słowo danych zajmuje więcej niż 1 bajt, powstaje problem, w jakiej kolejności zapisane są w pamięci kolejne jego bajty. Standardowym sposobem zapisu w procesorach ARM jest tzw. konwencja *little*



Rys. 1.5. Sposób mapowania adresów z obszaru *bit-band*



Rys. 1.6. Sposób zapisu danych w konwencjach *big endian i little endian*

endian, w której najmniej znaczący (najmłodszy) bajt jest zapisywany jako pierwszy, czyli pod najniższym adresem zajmowanego bloku pamięci. Ciekawą cechą procesorów ARM jest możliwość używania także konwencji *big endian*, w której jako pierwszy jest zapisywany bajt najbardziej znaczący. Różnice pomiędzy tymi sposobami zapisu pokazano także na **rysunku 1.6**. Wybór konwencji następuje w momencie startu lub zerowania procesora na podstawie wartości logicznej napięcia podanego na wyprowadzeniu BIGEND obudowy procesora. Zaletą stosowania konwencji *big endian* jest możliwość ułatwienia pisania i przyśpieszenia aplikacji, które służą np. do komunikacji przez sieć Ethernet, gdzie stosowany jest właśnie zapis *big endian*. Unika się wówczas konieczności nieustannych konwersji między oboma sposobami zapisu.

1.3.3. Podstawowe elementy rdzenia

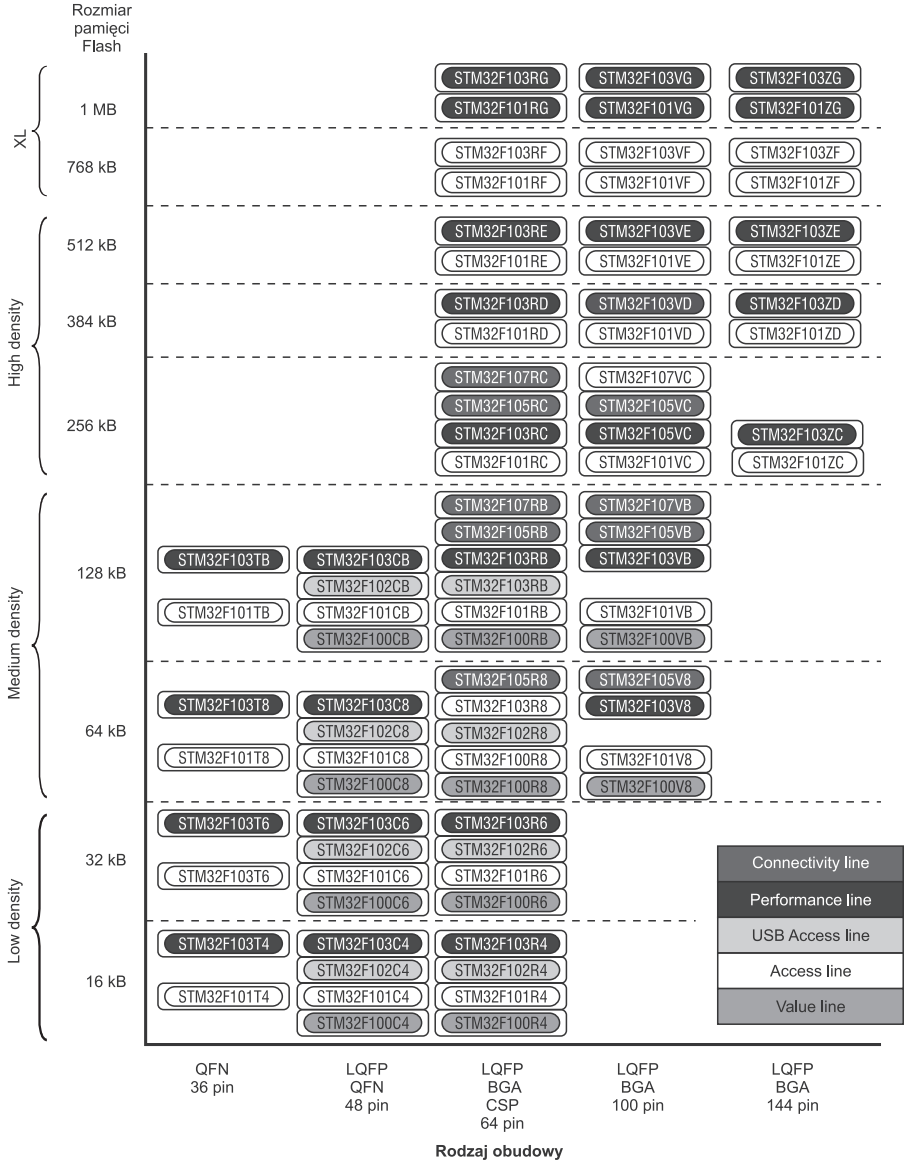
Oprócz jednostki arytmetyczno-logicznej będącej zasadniczym i najważniejszym elementem rdzenia, w jego skład wchodzą:

- interfejs pamięci,
- układ zarządzania zasilaniem i poborem energii,
- układy generowania sygnałów zegarowych dla poszczególnych podzespołów procesora,
- układ zarządzający magistralami systemowymi (*Bus Matrix*),
- kontroler przerwań (NVIC),
- układy śledzenia wykonania (debugowania) kodu i pracy krokowej,
- opcjonalny układ ochrony pamięci,
- interfejs JTAG/SW, pozwalający m.in. programować procesor w docelowym systemie oraz śledzić działanie programu w trybie debugowania.

Inne układy peryferyjne, jakie można znaleźć w różnych mikrokontrolerach pochodzących od poszczególnych wytwórców, nie należą do rdzenia i nie są dziełem firmy ARM. Są one natomiast dodawane do głównego rdzenia przez producentów mikrokontrolerów.

1.4. Mikrokontrolery STM32F10x

STMicroelectronics jest jednym z największych na świecie producentów układów elektronicznych. Posiada on fabryki i centra badawcze w Europie, Azji i Afryce. Jak przystało na tak znaczącego uczestnika rynku, ma w swojej ofercie mikrokontrolery 8- i 32-bitowe. Wśród nich najbogatszą grupę stanowią układy 32-bitowe wykorzystujące rdzenie ARM Cortex, których obecnie w ofercie jest ponad 100,



Rys. 1.7. Zestawienie oferowanych mikrokontrolerów z rodziny STM32F1

w różnych wersjach. Układy te noszą oznaczenie STM32F10xxx, gdzie xxx to oznaczenie konkretnego modelu z rodziny STM32. Zestawienie dostępnych wersji przedstawiono na **rysunku 1.7**.

Jak widać, układy STM32 oferowane są w pięciu rodzinach – liniach produktów:

– linia *Performance* to podstawowa linia układów,

– linia *Connectivity* to linia układów wyposażonych dodatkowo w interfejs Ethernet,

– linia *Access* to linia układów tańszych, zubożonych w porównaniu z linią podstawową przede wszystkim o układy interfejsów komunikacyjnych,
– linia *USB Access* to linia układów podobnych to tych z linii *Access*, ale z dodaną obsługą interfejsu USB,
– linia *Value* to linia układów najtańszych i najbardziej okrojonych pod względem wyposażenia.

Kolejnym kryterium podziału układów na rodziny jest rozmiar pamięci Flash. Firma STM używa w dokumentacjach terminu „gęstość" (*density*) na określenie pojemności pamięci zawartej w układzie. Bierze się to stąd, że im więcej pamięci ma zawierać mikrokontroler, tym bardziej musi być ona upakowana w chipie, a więc musi mieć większą gęstość.

Ostatnim kryterium podziału jest rodzaj obudowy, w jakiej jest umieszczony mikrokontroler. Najmniejsze i najuboższe pod względem wyposażenia układy można znaleźć w małych obudowach o 36 wyprowadzeniach. Z kolei najbogatsze w wyposażenie mikrokontrolery, z wieloma układami peryferyjnymi, oferowane są w obudowach ze 100, a nawet 144 wyprowadzeniami.

Duża różnorodność oferowanych wersji pozwala konstruktorom wybrać mikrokontroler najlepiej dostosowany do ich wymagań pod względem wyposażenia w pamięć i układy peryferyjne, rozmiarów fizycznych obudowy i ceny. Dobór odpowiedniego, najlepiej dopasowanego do projektowanego systemu, a jednocześnie jak najtańszego układu jest często sporym wyzwaniem. Ułatwieniem jest fakt, że ponieważ wszystkie układy wykorzystują rdzeń ARM Cortex, zachowana jest duża kompatybilność programowa i pełna kompatybilność fizyczna mikrokontrolerów montowanych w obudowach tego samego typu. Dzięki temu łatwo jest przenieść kod napisany dla jednej wersji mikrokontrolera na inną.