Systemy wbudowane

Przykłady kodu Assembler

Wstęp

- Slowo assembly w języku angielskim oznacza składać, montować
- W odniesieniu do języka programowania słowo to odnosi się do niskopoziomowego języka programowania urządzeń elektronicznych
- Językiem niskopoziomowym nazywa się taki język, w którym każde z poleceń ma bezpośrednie przełożenie (najlepiej jeden do jednego) na instrukcję maszynową danej architektury sprzętowej
- Każda architektura sprzętowa posiada swój zestaw rozkazów co oznacza, że język assemblera nie jest językiem uniwersalnym

Wstęp

- W assemblerze używa się tzw. mnemoników.
- Mnomonik to w pojęciu naukowym technika wspomagająca naukę mniej przyswajalnych przez ludzki mózg informacji poprzez zaprezentowanie ich/zamianę oryginalnej wartości na inną, prostszą do zapamiętania
- Mnemoniki w assemblerze mają za zadanie odpowiadać każdemu numerowi instrukcji bądź rozkazowi maszynowemu danego układu

Wstęp

- Mnemonikami zastępowane są także bezpośrednie adresy rejestrów, segmentów czy flag mikroukładów
- Niekiedy często wykonywane operacje (seria operacji maszynowych) zapisywane są pod jednym mnemonikiem (dla wygody programistów)
- Aplikacje pisane w assemblerze generalnie powinny być lepiej dopasowane do wskazanej architektury, a co za tym idzie efektywniej wykorzystywać jej możliwości
- Niestety obecne rozbudowanie niektórych układów powoduje, że tylko najlepsi inżynierowie danego układu są w stanie sprostać napisaniu dobrego kodu ASM

Assembler AVR

- Procesor posiada 135 rozkazów mnemonicznych
- Większość rozkazów wykonuje się w jednym takcie procesora
- Rozkazy jedynie typu RISC
- Rozkazy dla poszczególnych kontrolerów są podobne, jednak w poszczególnych modelach może być ich mniej/więcej
- AVR32 ma zupełnie inny zestaw rozkazów, który nie pasuje do zestawu AVR8

Ogólne oznaczenia w AVRASM2

Status Register (SREG)

- SREG - Rejestr stanu
- C - Flaga przeniesienia
- Z - Flaga zerowania
- N - Flaga ujemna
- V - Wskaźnik przepełnienia U2
- S - N ⊕ V (test znaku)
- H – przeniesienia połówki
- T – bit transferu używany przez instrukcje BLD (Bit LoaD) oraz BST (Bit STore)
- I – flaga ogólnego włączenia/wyłączenia przerwań

Ogólne oznaczenia AVRASM2

Rejestry i operandy

- Rd - rejestr Przeznaczenia (i źródła)
- Rr - rejestr Źródłowy
- R – Wynik wykonanej instrukcji
- K – stałe dane
- k – stały adres
- b – bit w pliku rejestru bądź rejestru We/Wy (3-bit)
- s – bit w Rejstrze Statusu (3-bit)
- X,Y,Z – pośredni adres rejestru (X=R27:R26, Y=R29:R28 oraz Z=R31:R30)
- A – lokalizacja adresu We/Wy
- q – Przemieszczenie dla bezpośredniowego adresowania (6-bit)

Rejestry Wejścia/Wyjścia

- RAMPX, RAMPY, RAMPZ – pozwalają na pośrednie adresowanie całej dostępnej przestrzeni kontrolera MCU (powyżej 64 KB) oraz wypełnianie danych w pamięci kontrolera powyżej 64 KB
- RAMPD – rejestr złączony do rejestrem Z pozwalający na bezpośrednie adresowanie przestrzeni danych kontrolera (powyżej 64KB)
- EIND – rejestr złączony z rejestrem Z pozwalający na pośredni skok i wywołanie w całej przestrzeni pamięci programu MCU (większego niż 128KB/64K słów)

Rejestry Wejścia/Wyjścia

Stos

- STACK – Stos dla zwortu adresu i przechowywania rejestrów
- SP – Wskaźnik Rejestru na STOS

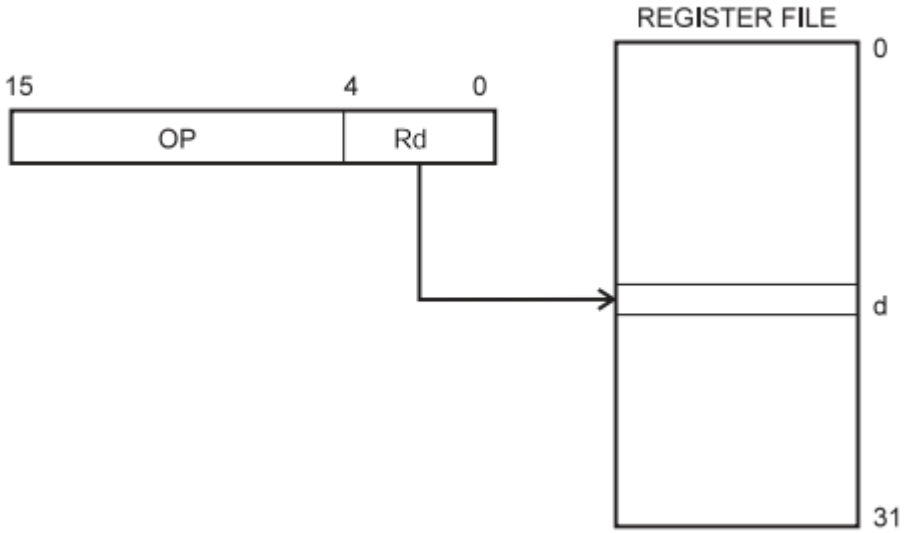
Flagi

- ⇔ – flaga zależna od instrukcji
- 0 – flaga czyszczona przez instrukcję
- 1 – flaga ustawiona przez instrukcję
- - – stan flagi nie zależy od instrukcji

Dostęp bezpośredni do rejestrów

Register Direct, Single Register Rd

Figure 2-1. Direct Single Register Addressing

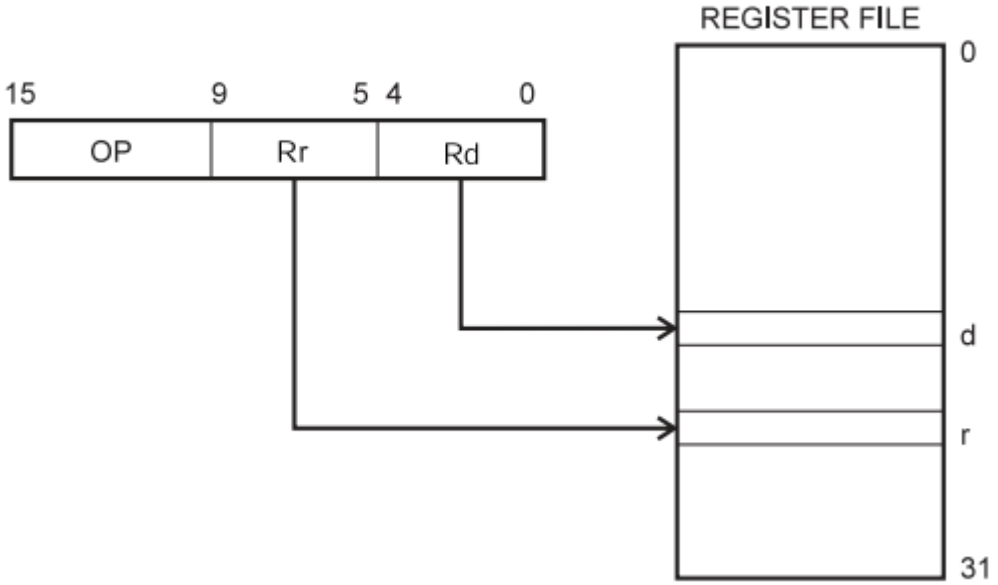


The operand is contained in register d (Rd).

Dostęp bezpośredni do rejestrów

Register Direct - Two Registers, Rd and Rr

Figure 2-2. Direct Register Addressing, Two Registers

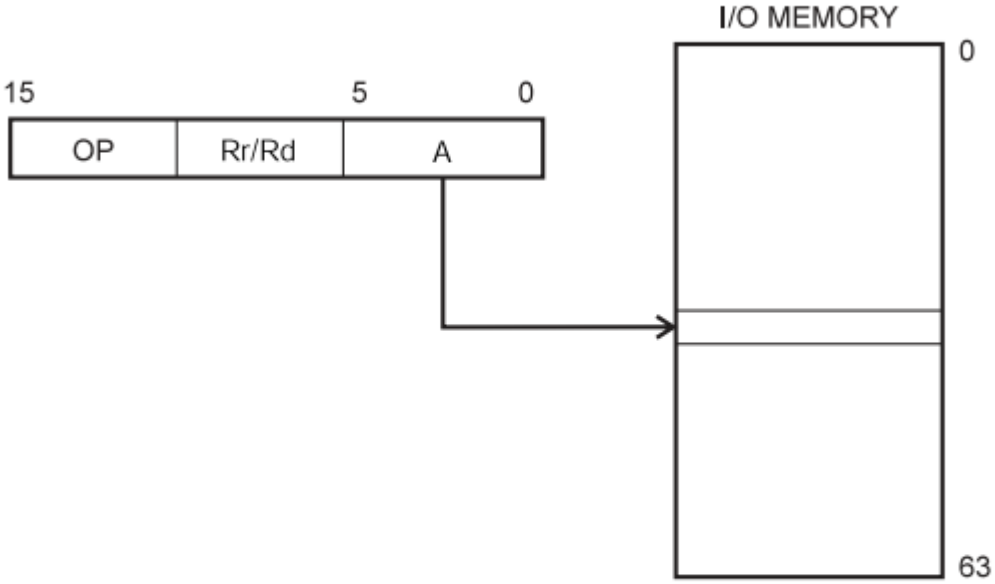


Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

Bezpośrednie Wejście/Wyjście

I/O Direct

Figure 2-3. I/O Direct Addressing



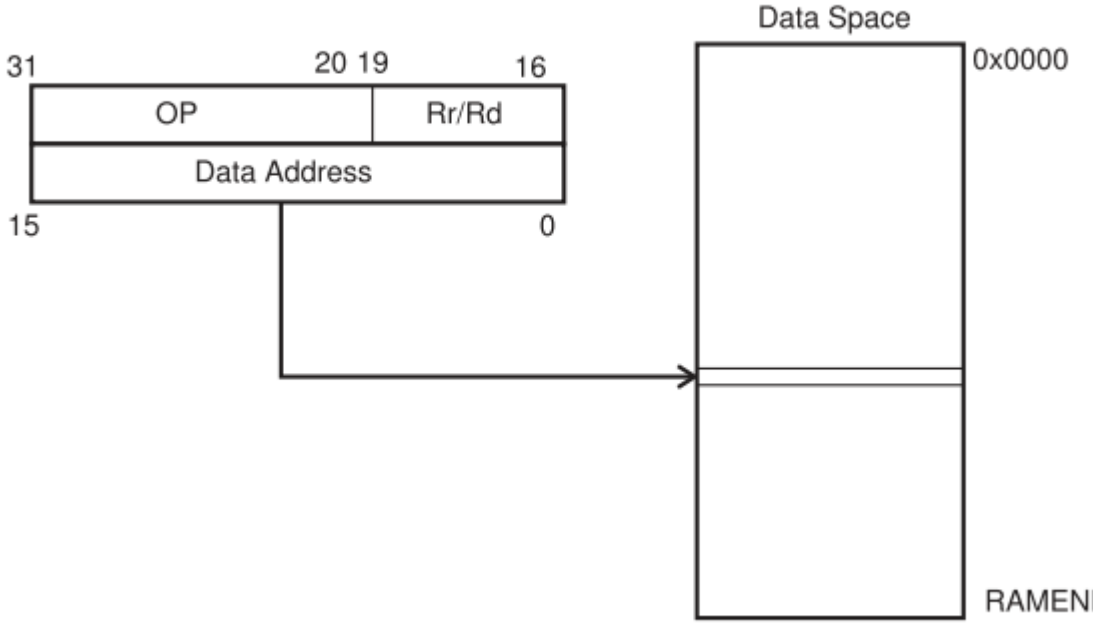
Operand address is contained in six bits of the instruction word. n is the destination or source register address.

Note: Some complex AVR Microcontrollers have more peripheral units than can be supported within the 64 locations reserved in the opcode for I/O direct addressing. The extended I/O memory from address 64 to 255 can only be reached by data addressing, not I/O addressing.

Bezpośrednie adresowanie danych

Data Direct

Figure 2-4. Direct Data Addressing

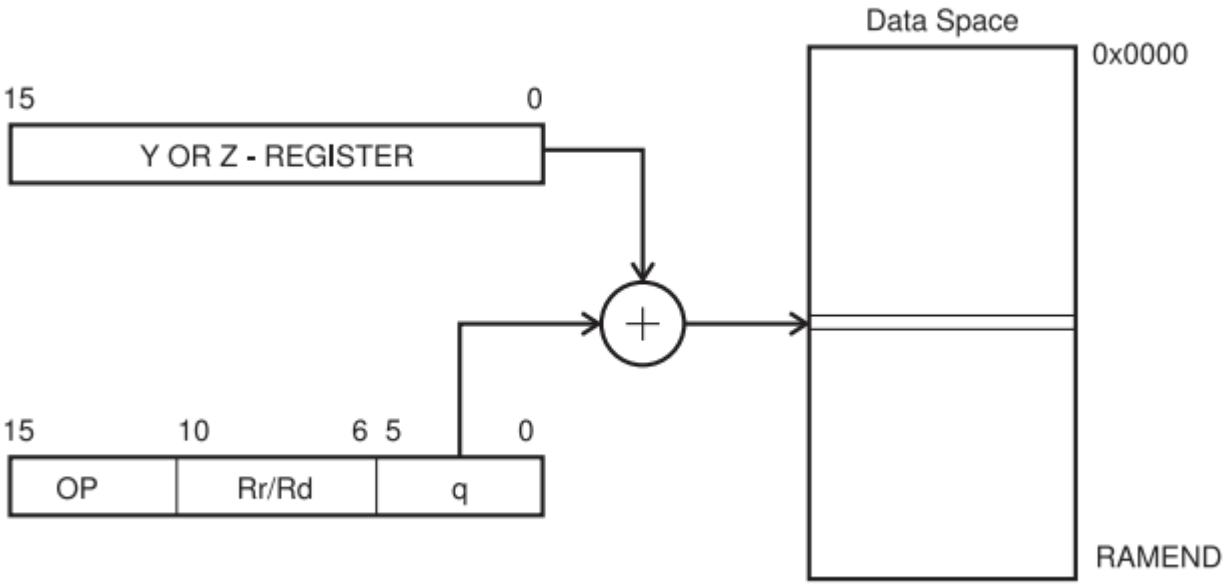


A 16-bit Data Address is contained in the 16 LSBs of a two-word instruction. Rd/Rr specify the destination or source register.

Pośrednie adresowanie z przeniesieniem

Data Indirect with Displacement

Figure 2-5. Data Indirect with Displacement

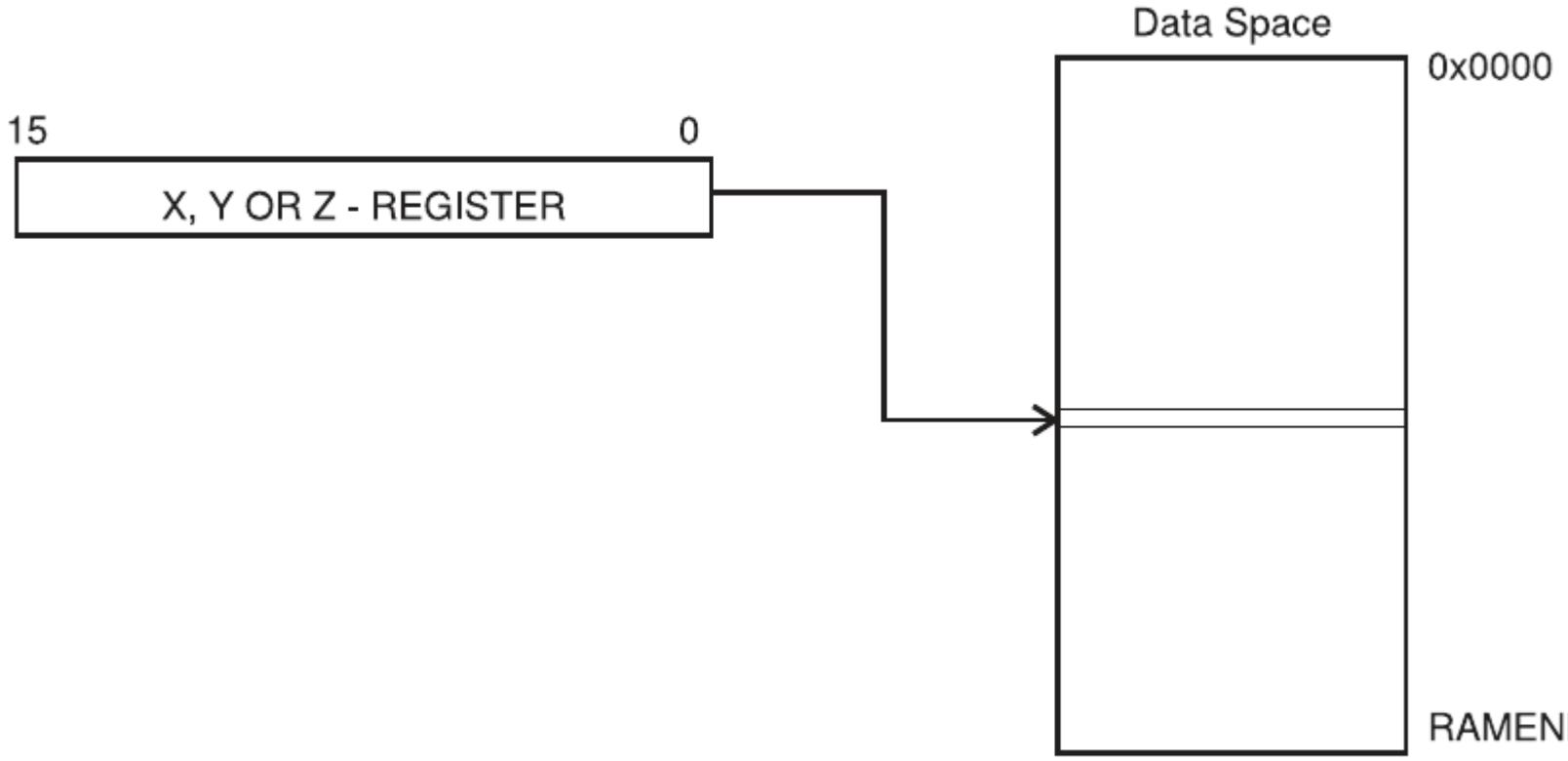


Operand address is the result of the Y- or Z-register contents added to the address contained in six bits of the instruction word. Rd/Rr specify the destination or source register.

Adresowanie pośrednie

Data Indirect

Figure 2-6. Data Indirect Addressing

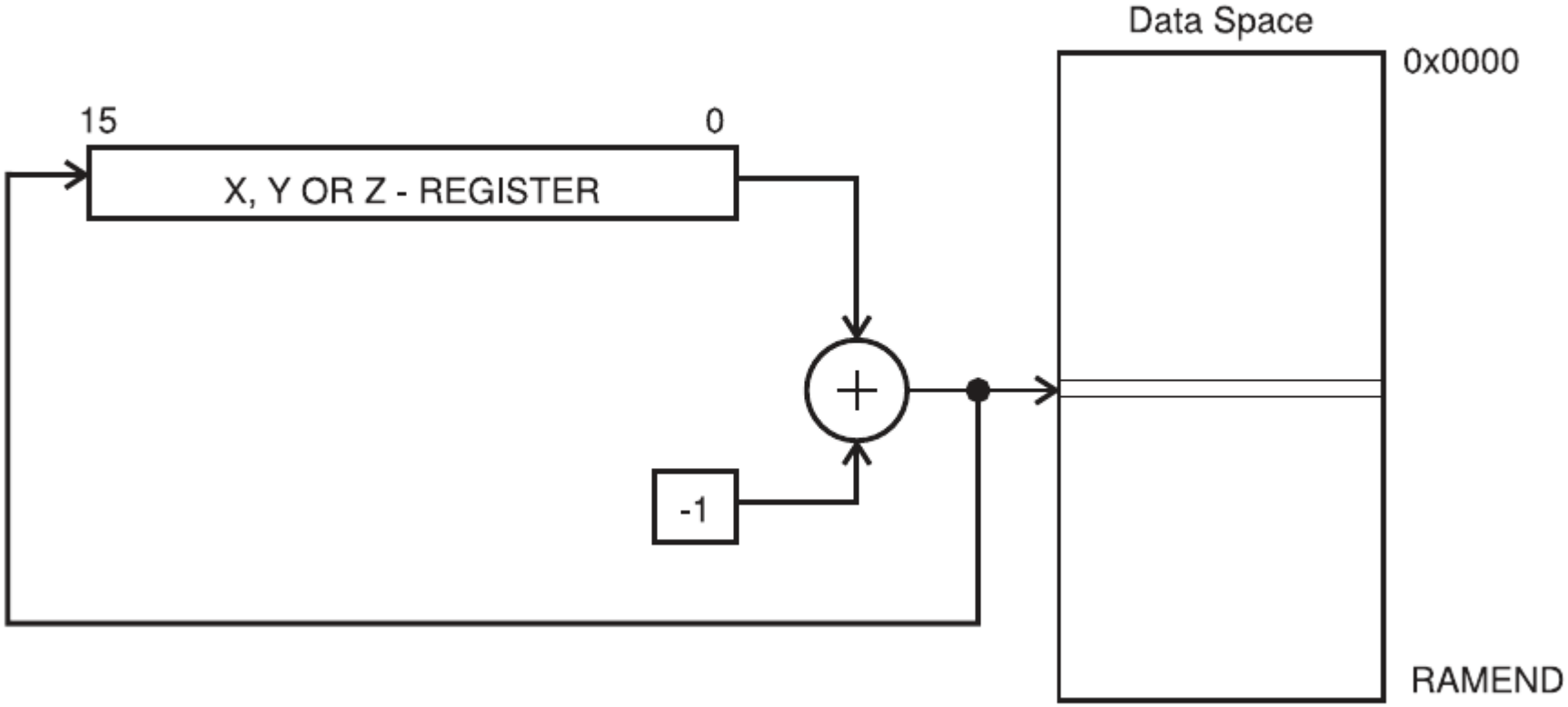


Operand address is the contents of the X-, Y-, or the Z-register. In AVR devices without SRAM, Data Indirect Addressing is called Register Indirect Addressing. Register Indirect Addressing is a subset of Data Indirect Addressing since the data space form 0 to 31 is the Register File.

Pośrednie adresowanie danych z obniżeniem wartości o 1

Data Indirect with Pre-decrement

Figure 2-7. Data Indirect Addressing with Pre-decrement

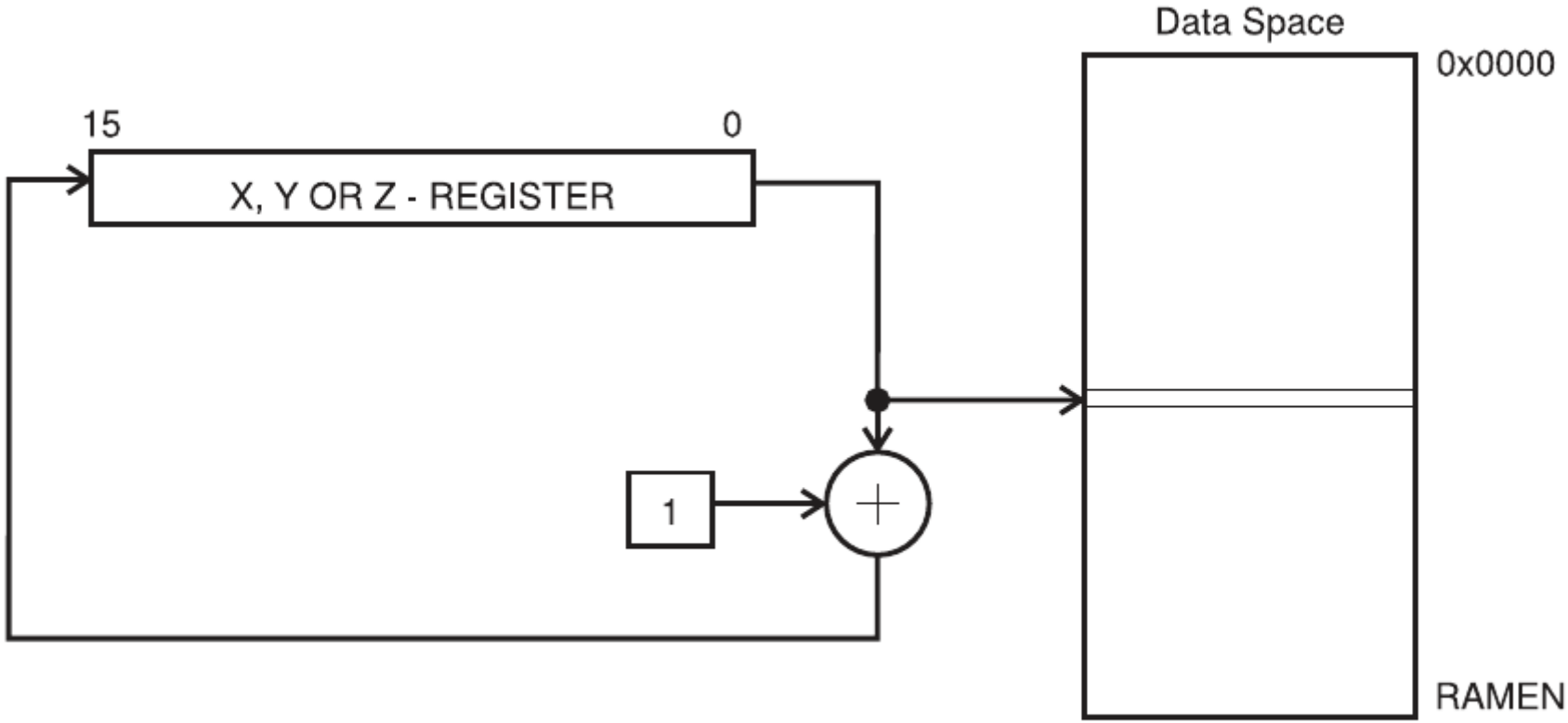


The X,- Y-, or the Z-register is decremented before the operation. Operand address is the decremented contents of the X-, Y-, or the Z-register.

Pośrednia adresacja z powiększeniem o jeden

Data Indirect with Post-increment

Figure 2-8. Data Indirect Addressing with Post-increment

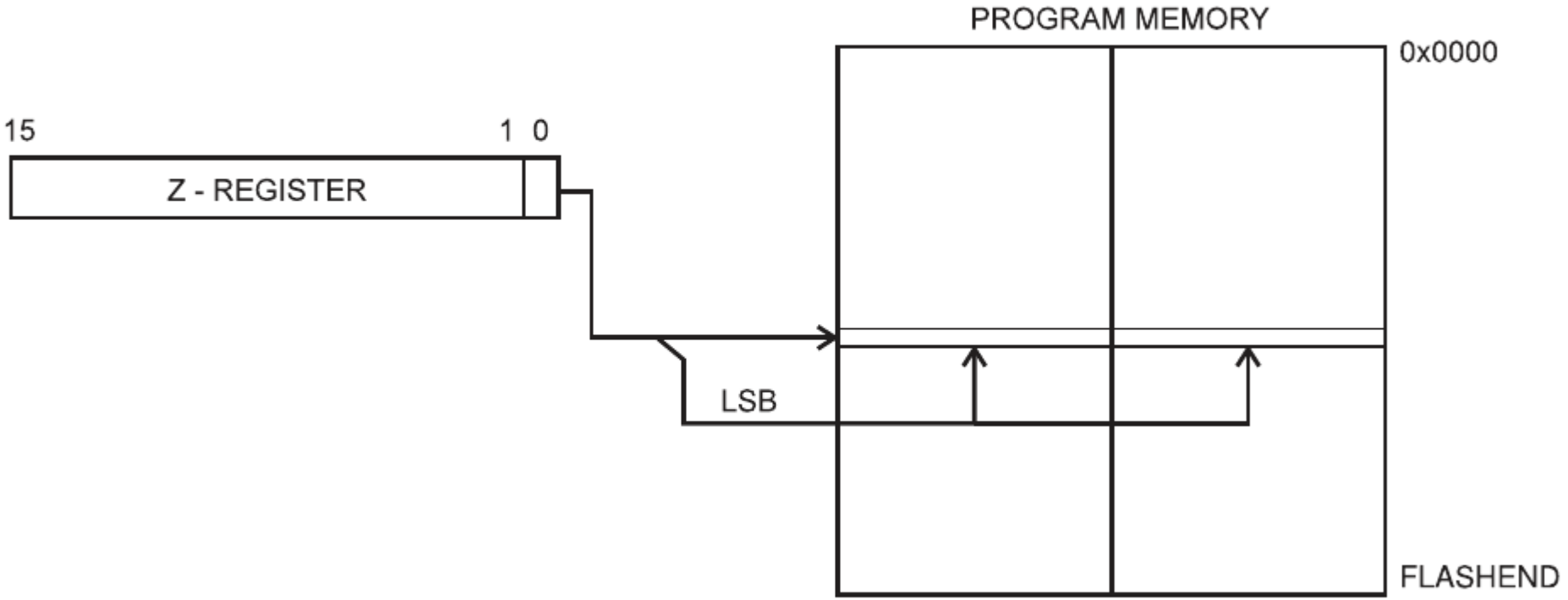


The X-, Y-, or the Z-register is incremented after the operation. Operand address is the content of the X-, Y-, or the Z-register prior to incrementing.

Stała adresacja pamięci programu przy użyciu Load Program Memory, Extended Load Program Memory oraz Storage Program Memory

Program Memory Constant Addressing using the LPM, ELPM, and SPM Instructions

Figure 2-9. Program Memory Constant Addressing

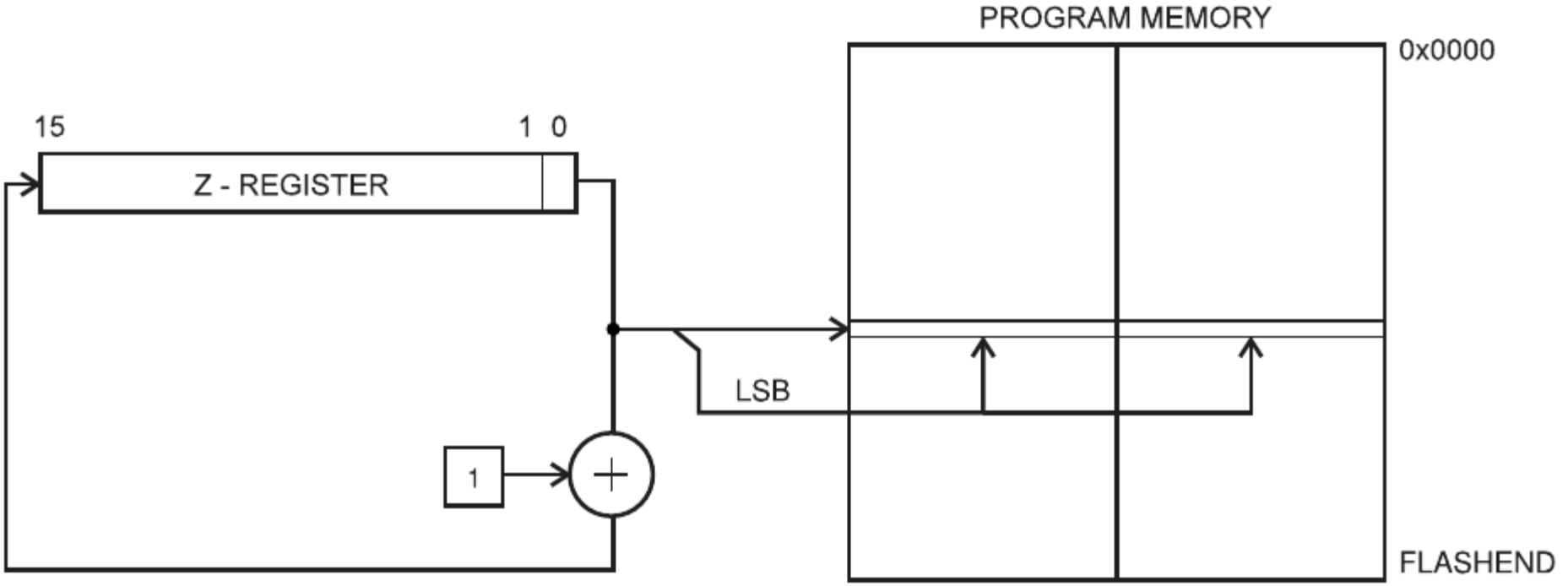


Constant byte address is specified by the Z-register contents. The 15 MSBs select word address. For LPM, the LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1). For SPM, the LSB should be cleared. If ELPM is used, the RAMPZ Register is used to extend the Z-register.

Adresacja programu z inkrementacją

Program Memory with Post-increment using the LPM Z+ and ELPM Z+ Instruction

Figure 2-10. Program Memory Addressing with Post-increment

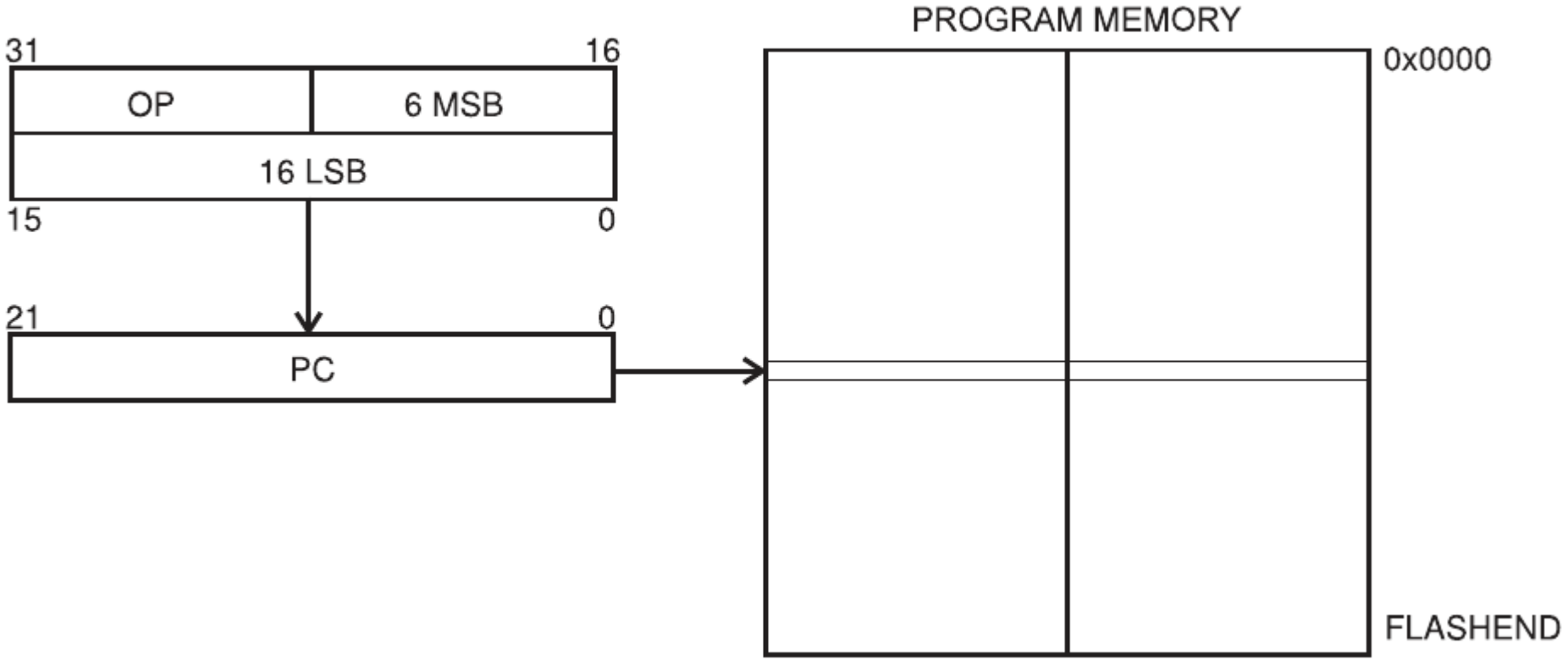


Constant byte address is specified by the Z-register contents. The 15 MSBs select word address. The LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1). If ELPM Z+ is used, the RAMPZ Register is used to extend the Z-register.

Bezpośrednia adresacja programu poprzez skok i wywołanie

Direct Program Addressing, JMP and CALL

Figure 2-11. Direct Program Memory Addressing

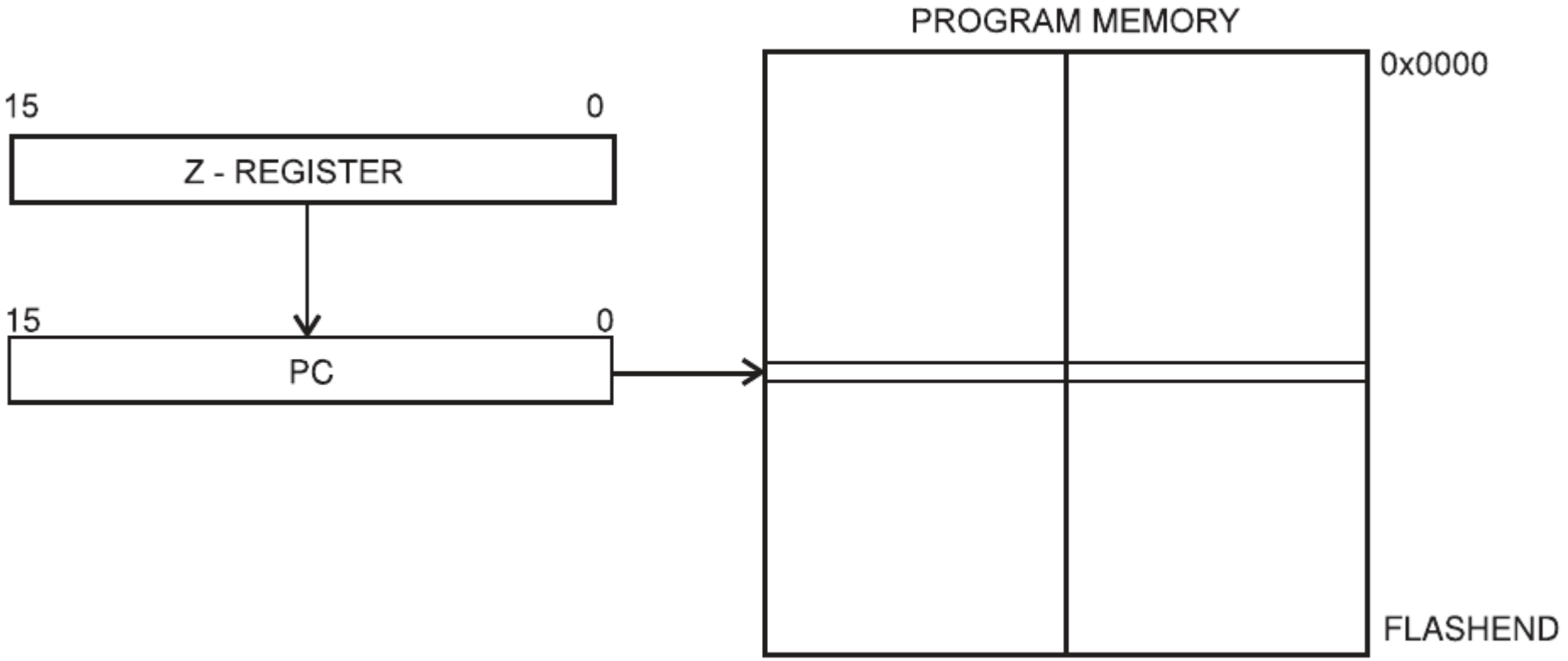


Program execution continues at the address immediate in the instruction word.

Pośrednie adresowanie pamięci programu

Indirect Program Addressing, IJMP and ICALL

Figure 2-12. Indirect Program Memory Addressing

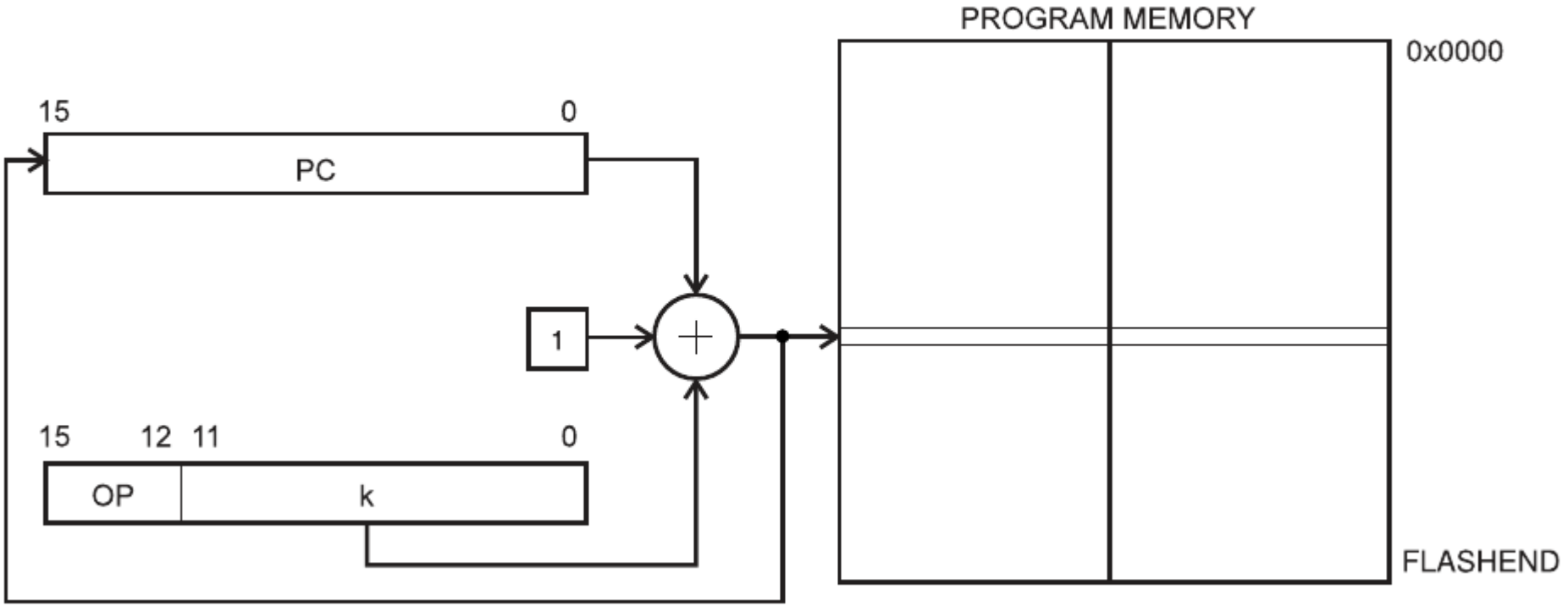


Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the contents of the Z-register).

Względna adresacja pamięci programowej

Relative Program Addressing, RJMP and RCALL

Figure 2-13. Relative Program Memory Addressing



Program execution continues at address PC + k + 1. The relative address k is from -2048 to 2047.

Materiały

- https://en.wikipedia.org/wiki/Assembly_language
- https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.asma400/asmr102112.htm
- http://www.atmel.com/webdoc/avrassembler/index.html
- https://www.renesas.com/en-us/support/technical-resources/engineer-school/mcu-01-basic-structure-operation.html
- http://www.atmel.com/images/Atmel-0856-AVR-Instruction-Set-Manual.pdf