

<p>Adresacja pośrednia, zwana adresacją rejestrową, tryb adresacji prostej (jednoskładnikowej), gdzie pojedyncze słowo binarne wskazuje lokalizację operandu. Adresowanie pośrednie polega na zapisaniu w kodzie rozkazu adresu komórki pamięci, w którym znajduje się już właściwy adres, czyli zawieramy w kodzie rozkazu umowny adres lub nazwę rejestru. Przykład: MVPD 0x1000, *AR5- ; Operand *AR5- jest właśnie przykładem adresacji pośredniej</p> <p>Adresowanie bezpośrednie (direct) - tryb adresowania (czyli sposób wskazywania na argumenty wykorzystywane w trakcie wykonywania instrukcji) odnoszący się do instrukcji, w których po kodzie rozkazu następuje adres argumentu umieszczonego w pamięci danych (komórki pamięci RAM), a zatem położenie operandu jest określane poprzez podanie w kodzie rozkazu jawnego, pełnego adresu w przestrzeni adresowej. Przykład: MOV 4Ah, 8 - wykonując ten rozkaz, mikrokontroler pobiera zawartość komórki o adresie 8 i przesyła ją do bitowym adresem, gdy spełniony jest prosty lub złożony warunek (Zazwyczaj wykorzystywane do testów arytmetycznych wykonywanych na zawartości akumulatora lub testowania flag. – BANA: przeładowuje PC, jeżeli zawartość wybranego rejestru pomocniczego AR nie jest równa zero. Rejestr ten wykorzystywany jest jako licznik pętli gdyż rozkaz BANA poza sprawdzaniem wykonuje również jego dekrementację. Wykorzystuje się to w szczególności do implementacji pętli FOR. - instrukcje skoku bezwarunkowego – dla pętli bez końca. - w obsłudze pętli: rozkazy skoków czy repetycji bloków muszą zawierać etykiety wskazujące zakres skoku / pętli</p> <p>Wait State Generator - programowy generator cykli oczekiwania. Jeśli np. pamięć flash pracuje z dość ograniczoną prędkością ok 30 Mhz, a CPU o wiele szybciej, w szczególnych przypadkach konieczne jest opóźnienie dostępu do flasha. Właśnie w takich sytuacjach wykorzystuje się Wait State</p> <p>Bootowaniem (ang. booting) proces bootstrappingu prowadzący do uruchomienia systemu operacyjnego po włączeniu komputera. Wszystkie komputery (jako sprzęt) potrafią uruchamiać tylko te programy, które znajdują się w głównej pamięci. bootloadery (programy rozruchowe), realizują ideę bootstrappingu. bootloder nie posiada pełnej funkcjonalności systemu operacyjnego, ale potrafi załadować taką jego część, która pozwoli na jego całkowite uruchomienie</p> <p>HOLD - Sygnał wejściowy, pozwalający zewnętrznym urządzeniom na żądanie przejęcia kontroli nad magistralami. Stan aktywny na wyjściu procesora HLDA (hold acknowledge) sygnalizuje przejście w stan zawieszenia. Wstrzymanie pracy procesora może być wykonane dopiero po zakończeniu cyklu przesłania. Po zakończeniu transmisji sterownik DMA zwraca procesorowi kontrolę nad magistralami.</p> <p>Rozkaz HALT (stan zatrzymania) powoduje zatrzymanie cyklu rozkazowego mikroprocesora z zachowaniem stanu wszystkich rejestrów. Jest on użyteczny przy poszukiwaniu błędów i poprawianiu programu</p> <p>Co to są sekcje programu i do czego są używane? Seksje to fragmenty programu zawierające jednorodne obiekty; kod, stałe, zmienne lub układy we/wy. Są one zdefiniowane za pomocą dyrektyw w zbiorach źródłowych. Seksje dzielimy wg. zawartości na; - sekcja inicjalizowana (kod programu, predefiniowane stałe), - sekcja nieinicjalizowana (rezerwacja obszarów pamięci na zmienne czy stałe) i wg. opisu na - sekcja nazwana (opatrzona nazwą) - sekcja nienazwana (bez nazwy) Seksje są umieszczane przez linker we wskazanych obszarach pamięci zgodnie z zapisem zbioru konfiguracyjnego. Seksje o tych samych nazwach łączone są we wspólne obszary ułatwiając organizację danych w pamięci.</p>	<p>Sposoby realizacji pętli i stosowane tam rozkazy. Do realizacji pętli mogą zostać wykorzystane następujące rozwiązania: - repetycja pojedynczego rozkazu realizowana instrukcją RPT lub RPTZ, pozwala na powtórzenie instrukcji od 1 do 65536 razy. Różnica między RPT a RPTZ polega na tym, że instrukcja RPTZ resetuje wskazany w rozkazie akumulator A lub B przez rozpoczęciem pętli. RPT n -> n+1 powtórzeń - repetycja bloku rozkazów realizowana za pomocą instrukcji RPTB. Pozwala ona na powtórzenie bloku instrukcji od 1 do 65536 razy. Liczbę obiegów pętli ustala się przez zawartość BRC (Block Repeat Counter) plus jeden. BRC=n -> n+1 powtórzeń - instrukcje skoku warunkowego, wykonujące skok tylko wtedy, gdy spełniony jest dany prosty lub złożony warunek (w przeciwnym razie wykonanie programu przechodzi do następnej instrukcji). Wyróżniamy dwie instrukcje skoku warunkowego: BC: przeładowuje PC bezpośrednim 16-bitowym adresem, gdy spełniony jest prosty lub złożony warunek. Zazwyczaj wykorzystywane do testów arytmetycznych wykonywanych na zawartości akumulatora lub testowania flag. - BANA: przeładowuje PC, jeżeli zawartość wybranego rejestru pomocniczego AR nie jest równa zero. Rejestr ten wykorzystywany jest jako licznik pętli gdyż rozkaz BANA poza sprawdzaniem wykonuje również jego dekrementację. Wykorzystuje się to w szczególności do implementacji pętli FOR. - instrukcje skoku bezwarunkowego – dla pętli bez końca. - w obsłudze pętli: rozkazy skoków czy repetycji bloków muszą zawierać etykiety wskazujące zakres skoku / pętli</p> <p>Co w procesorach określa pojęcie rozszerzenia znakowego i dlaczego jest ono tak istotne w DSP? Procesory sygnałowe dla zachowania odpowiedniej precyzji obliczeń zwykle dysponują akumulatorami co najmniej dwukrotnie większymi od rozmiaru słowa, którym pracują. W przypadku rodziny procesorów C5000 pracującej na słowie 16-to bitowym akumulatory mają rozmiar 40-to bitowy. Rozszerzenie znakowe to mechanizm pozwalający procesorowi w takiej sytuacji na zachowanie znaku danej ładowanej do większego rejestru. Operacja ta realizowana jest automatycznie i może być włączana za pomocą bitu SXM – Sign eXtention Mode – umieszczonego w rejestrze statusowym ST1. Zasada działania SXM jest następująca: SXM = 1 □ liczby ujemne dopełniane są na starszych bitach jedynkami, zaś dodatnie zerami. SXM = 0 □ brak dopełnienia znakowego, starsze bity pozostają zwykle bez zmian. Obsługa rozszerzenia znakowego wygląda następująco: SSBX SXM ;sign-extension mode ON RSBX SXM ;sign-extension mode OFF</p> <p>Do czego służy w procesorach DSP zegar (timer)? Zegary (timery) w DSP można zastosować do: ? generację przerwań po ustalonym programowo czasie (np. dla RTC) ? generowania impulsów zewnętrznych po ustalonym programowo czasie ? sterowania generacją impulsów PWM ? realizacji przetwornika C/A ? pomiaru czasu trwania funkcji czy innych procesów software'owych ? generację impulsów i pomiar ich szerokości ? generacji zdarzeń synchronizujących dla DMA, A/C, C/A i innymi peryferiami.</p>	<p>Wymień czynniki decydujące o szybkości realizacji programu w DSP. a) wynikające z budowy procesora □ częstotliwość taktowania procesora □ przetwarzanie nakładkowe □ zwielokrotnienie magistral □ rozkazy specjalizowane i ukierunkowane na aplikacje □ zastosowanie adresacji kołowej lub z odwracaniem bitów □ rozkazy skoków z opóźnieniem □ łączone warunki dla skoków i operacji warunkowych □ wykonywanie rozkazów w trybie repetycji □ zaawansowana obsługa pośrednich wyników operacji □ operacje dwusłowe □ wielkość pamięci wewnętrznej, szczególnie DARAM □ ilość i sposób wykorzystania przerwań oraz ich ewentualne kolizje z trybami repetycji b) wynikające ze sposobu przygotowania programu □ wykorzystanie wymienionych wyżej możliwości sprzętowych □ podział programu pomiędzy assembler i języki wysokiego poziomu □ rozmieszczenie danych w pamięciach SARAM / DARAM, pamięci zewnętrznej i/lub wewnętrznej</p> <p>Dyrektywa assemblera jest to polecenie definiujące mu sposób traktowania danego fragmentu programu. Są elementem sterowania asemblacją programu. Nie są tłumaczone na rozkazy programu a jedynie uruchamiają sposób działania assemblera. Dyrektywy mogą służyć np. do zdefiniowania sekcji w zbiorach źródłowych, uaktywnienia własności assemblera, itd. Są one poleceniami tekstowymi i zaczynają się od kropki. Przykładami dyrektyw mogą być: .mmregs włącza predefiniowane nazwy rejestrów MMR .sect „kot” kończy poprzednio zdefiniowaną sekcję i otwiera nową inicjalizowaną i nazwaną „kot” sekcję na kod programu lub dane .text kończy poprzednio zdefiniowaną sekcję i otwiera nową inicjalizowaną i nazwaną sekcję na kod programu .bss test,n kończy poprzednio zdefiniowaną sekcję i otwiera nową nieinicjalizowaną sekcję o nazwie test na n słów danych .usect „pies”, n kończy poprzednio zdefiniowaną sekcję i otwiera nową nieinicjalizowaną i nazwaną „pies” sekcję dla danych rezerwując dla nich n słów w pamięci danych tab .word 4, 0x13, 66h kończy poprzednio zdefiniowaną sekcję i otwiera nową inicjalizowaną sekcję dla trzech wartości 4, 13h, 66h zaczynającą się od adresu „tab”</p> <p>Co to jest i czemu służy w procesorach rodziny C54xx IPTR? IPTR (czyli Interrupt Pointer) to 9-cio bitowy rejestr, fragment rejestru PMST. Jego zawartość stanowi najstarsze 9 bitów adresu w tablicy wektorów przerwań. Uzupełniona kodowanym na 5 bitach numerem przerwania i najmłodszymi dwoma bitami 00 tworzy adres początkowy w tablicy wektorów przerwań procesora. Te 9 bitów uzupełnione 7-mioma zerami tworzy adres położenia pierwszego wektora w tej tablicy. Jest to wektor przerwania RESET złożony z pierwszych czterech słów programu jego obsługi – startu procesora. Po sprzętowym RESET procesora IPTR = 1 1111 1111B = 0x1FF co lokuje początek tej tablicy pod adresem 0xFF80 (bo jego zawartość umieszczana jest na najstarszych bitach uprzednio wyzerowanego rejestru PC). Programista może przełączyć procesor do odczytywania tablicy wektorów przerwań z innego miejsca w pamięci programu przez zmianę zawartości IPTR i wykonanie programowego RESET.</p>	<p>Co wiąże, a co różni indywidualną maskę przerwania i flagę przerwania? I flaga i maska są występującymi w różnych rejestrach bitami (przerzutnikami). Wiąże je to samo przerwanie, tyle że maska jest bitem blokującym lub dopuszczającym obsługę przerwania a flaga jest bitem zgłoszenia rządania obsługi przerwania. Od strony operacyjnej łączy pewne podobieństwo. Flagą jest ustawiana sprzętowo za sprawą wystąpienia zewnętrznego sygnału, choć może być również ustawiona programowo. Jednak kasowana jest WYŁĄCZNIE sprzętowo na początku obsługi przerwania i po RESET. Zaś maska może być ustawiana i kasowana programowo a dodatkowo ustawiana jest sprzętowo przy rozpoczęciu obsługi przerwania i po RESET.</p> <p>Co odróżnia standardowy port szeregowy od McBSP w C54xx? McBSP to wielokanałowy buforowany port szeregowy. Od standardowego portu szeregowego odróżniają go następujące cechy: □ pełny, dwukierunkowy bezpośredni interfejs do układu codec i innych urządzeń szeregowych. □ podwójne buforowanie dla nadawania i potrójne dla odbioru transmisji □ zdolność realizacji wielu standardów komunikacji szeregowej □ praca wielokanałowa maksymalnie do 128 kanałów □ długość słowa: 8-, 12-, 16-, 20-, 24-, 32-bit □ wewnętrzna generacja zegara/ramek z SGR (Sample Rate Generator) □ transmisja 8-bitowa danych z możliwością wyboru pierwszeństwa LSB lub MSB Ponadto, podobnie jak standardowy port szeregowy, McBSP zapewnia: □ maksymalna szybkość bitowa: 1/2 CPU Clock Rate □ wbudowany komparing wg. praw μ lub A □ programowana polaryzacja zegara / ramek □ potrafi sygnalizować wszystkie typowe błędy i statusy</p> <p>Co to jest DARAM i dlaczego jest korzystna w procesorach DSP C54xx? DARAM (Double Access RAM) jest to pamięć zezwalająca na 2 dostępy w jednym cyklu procesora w każdym z bloków pamięci. Oznacza to, że zarówno CPU jak i peryferia mogą dokonywać odczytu i zapisu w tym samym cyklu. Część rozkazów może być efektywnie wykonywana tylko gdy ich operandy rozmieszczone są w DARAM (np. rozkazy z grupy MAC). Istotne jest ponadto, że pamięć DARAM z przestrzeni pamięci danych można przełączyć (uwidocznic) do przestrzeni pamięci programu. Służy do tego odpowiednie ustawienie bitu OVLY.</p> <p>Jak rozumiesz i co określa pojęcie trybu adresacji? Tryb adresacji jest sposobem definiowania dostępu do operandu w treści rozkazu (podawania adresu w rozkazie). Określa on, w jaki sposób instrukcje sięgają do swoich operandów w pamięci. Wyróżniamy następujące tryby adresacji: - natychmiastowy np. LD #10,A - absolutny LD A, *(y) - akumulatorem READ A x - pośredni LD *AR1,A - bezpośredni LD @x,A - za pomocą wskaźnika stosu PSHM ST1 - MMR LDM ST1,B Tryby zdresacji i ich rozumienie i sprawność wykorzystania to kluczowy element efektywnego programowania przetwarzania we wszystkich procesorach nie tylko sygnałowych.</p>
---	---	---	---

<p>Co to jest przetwarzanie nakładkowe, na czym polega i czemu służy? Przetwarzanie nakładkowe, albo kolejka (zargonowo pipelining), jest to sposób wykorzystania zasobów procesora do realizacji rozkazów tak, by żaden jego fragment „nie stał bezczynnie”. Uwarunkowane jest podziałem realizacji rozkazu na kolejne fazy wykonywane w pojedynczych cyklach rozkazowych i możliwościami bloków przetwarzających procesora oraz magistral transportu danych i rozkazów. Polega on na na równoczesnym wykonywaniu różnych faz kolejnych rozkazów programu. Przykładowo, wykonując fazę Pre-Fetch dla jednej instrukcji, procesor może jednocześnie wykonywać fazę Fetch poprzedniej instrukcji, fazę Decode dla jeszcze wcześniejszej instrukcji itd. Dzięki temu rozkazy pobierane do kolejki są wykonywane quasi równolegle – po jednej fazie z każdego z kolejnych rozkazów – jak gdyby cały jeden rozkaz w jednej fazie. Technika ta nie skraca czasu wykonywania pojedynczego rozkazu ale dzięki nakładaniu na siebie rozkazów pozwala na skrócenie wykonywania sekwencji rozkazów i przyspieszenie wykonania całego programu. W przetwarzaniu kolejkowym pobierając zawartość kolejnych komórek z pamięci programu natrafia się na trudności związane z wykonywaniem skoków. Typowy skok przerywa ciągłość kolejki uniemożliwiając wykorzystanie wszystkich rozkazów pobranych do kolejki. (Jeśli wykonujemy skok nie ma uzasadnienia do wykonywania rozkazów po skoku.) Przeciwdziałać tym trudnościom można przez specyficzną modyfikację programu polegającą na</p> <ol style="list-style-type: none"> 1. wykorzystaniu rozkazów skoków z opóźnieniem (np. BD, RD, ...) 2. przestawieniu niektórych rozkazów sprzed rozkazu skoku (np. B, R, ...) za rozkazy (np. BD, RD, ...). <p>Objaśnij zadania linkera w środowisku programów do generacji kodu procesora DSP. Linker łączy plik *.obj i generuje docelowy plik wyjściowy *.out. Rozmieszcza on i łączy jednoimienne sekcje w obszarach pamięci wskazanych w zbiorze/poleceniach konfiguracyjnych. Linker może generować różne, pomocne w analizie i uruchamianiu programu zbiory np. *.map – mapę pamięci, *.lst – pełnego listingu programu, *.hex – zbiór dla programatora pamięci, itd. Zajmuje się on rozmieszczeniem relokowalnych zbiorów *.obj a w nich symboli i sekcji, by przypisać je do ostatecznych adresów oraz decyduje o zewnętrznych powiązaniach między plikami wejściowymi i bibliotekami. Do prawidłowego działania linkera niezbędny jest zbiór konfiguracyjny linkera - Linker Command File (w CCS ma on rozszerzenie *.cmd).</p> <p>Do czego są szczególnie przydatne w procesorach DSP kanały DMA i z czym głównie współpracują? Kanały DMA w procesorach DSP współpracują głównie z McBSP oraz D/A. Można mówić o ich szczególnej przydatności ze względu na to, że:</p> <ul style="list-style-type: none"> ? DMA dla przesłań może sięgać do każdego zasobu ? prowadzą transfer danych bez zaangażowania CPU, a zatem odciążają procesor ? posiadają autolimit (automatyczne ustawienie następnego kanału do transferu) ? transfer można synchronizować np. z 20 zdarzeniami w C55xx ? każdy z kanałów dysponuje FIFO by zapis mógł wyprzedzać odczyt (gdy zasoby docelowe są zajęte) ? przy wydzieleniu kanału z użyciem DMA obsługiwanych może być do 128 kanałów ? pozwala na niezależny wybór wielu kanałów (słów), które mają być transmitowane i odbierane ? elastycznie indeksowana adresacja DMA pozwala na sortowanie każdego kanału do oddzielnego bufora 	<p>Od czego można uzależnić przebieg programu w procesorach rodziny C54xx? Generalnie sekwencyjny przebieg rozkazu modyfikują skoki. W tym skoki warunkowe mają szczególne znaczenie, bo realizowane są w zależności od spełnienia lub nie warunku lub warunków. Pytanie dotyczy wskazania od czego można uzależnić przebieg programu czyli jakie warunki jesteśmy w stanie wykorzystać w tych warunkowych skokach, A zatem Wiele stanów jest wykrywane i sygnalizowane flagami;</p> <ul style="list-style-type: none"> • OVA i OVB czyli przekroczenia w każdym z akumulatorów • C – Carry – przeniesienie w użytym akumulatorze, • TC – bit, gdzie trafiają wyniki operacji logicznych <p>Inne elementy mogą być wykrywane bezpośrednio (zwykle komparatorami);</p> <ul style="list-style-type: none"> • Zawartość akumulatora i jej relacja względem zera, • Relacja między zawartościami obu akumulatorów, • Zawartość rejestru adresowego i jego wartość zerowa, • Stan wejścia sygnałowego BIO • Stan testowanego dowolnego bitu w pamięci danych (trafi do TC) • Zawartość całej komórki w pamięci danych • Iloczyn do trzech warunków równocześnie w rozkazach warunkowych <p>59.Dlaczego w procesorach sygnałowych rejestr akumulatora jest ponad dwa razy większy od rozmiaru słowa, jakim pracują? Aby przyjąć wynik mnożenia liczb binarnych, potrzebny jest akumulator będący dwukrotnie większy niż rozmiar słowa. Wynik mnożenia odbierany do przechowania w pamięci danych znajduje się w takiej sytuacji na starszej części akumulatora – AH. Młodsza część akumulatora – AL młodsze 16 bitów – ma za zadanie zapewnienie większej rozdzielczości dla sumowania wyników mnożeń i uniknięcia kumulowania błędów na skutek przedwczesnego ograniczania rozdzielczości reprezentacji. Ta młodsza część akumulatora może uzyskać swój wpływ na wartość wyniku poprzez operację zaokrąglania (Rounding). Z kolei dodawanie może doprowadzić do wyniku wykraczającego ponad 32 bity akumulatora stąd rezerwę do sumowania lub wyników pośrednich sumowania zapewniają bity AG - GUARD (jest ich osiem).</p> <p>60.Co to są tryby repetycji i czemu służą w procesorach DSP rodziny C54xx? Tryb repetycji polega na powtarzaniu rozkazu lub bloku rozkazów. W trybie tym dzięki sprzętowej obłudze licznika pętli nie tracimy czasu na rozkazy sprawdzające licznik i realizujące skok. Stąd pętle takie są bardziej efektywne, tylko użyteczna część pętli zajmuje czas wykonania.Repetycja pojedynczego rozkazu:</p> <ul style="list-style-type: none"> - uruchamiana instrukcją RPT lub RPTZ, pozwala na powtórzenie następnej instrukcji od 1 do 65536 razy. Różnica między RPT a RPTZ polega na tym, że instrukcja RPTZ resetuje akumulator A lub B przez rozpoczęciem pętli. RPT n A n+1 powtórzeń. Niestety pętli takiej nie można przerwać przerwaniem! <p>Repetycja bloku rozkazów:</p> <ul style="list-style-type: none"> - uruchamiana za pomocą instrukcji RPTB „etykieta” albo z zastosowaniem mechanizmu opóźnień RPTBD „etykieta”. <p>Pozwala na powtórzenie od 1 do 65536 razy bloku instrukcji od inicjującego rozkazu do rozkazu opatrzonego etykietą. Liczba przebiegów zadana jest zawartością BRC (Block Repeat Counter) plus jeden; BRC=n A n+1 powtórzeń. Zakres pętli zadawany jest zawartościami rejestrów RSA – adres początku pętli, REA- adres końca pętli.</p>	<p>41. Co to jest procedura obsługi przerwania i jakie są jej główne cechy? Procedura obsługi przerwania (ISR) to przygotowany fragment programu zawierający sekwencję rozkazów opisujących sposób reagowania procesora (systemu) na występujące zdarzenie – jego sygnał. Dla prawidłowego działania procedury obsługi przerwania należy poza jej przygotowaniem ustawić globalną maskę przerw – INTM, odpowiedni bit indywidualnej maski danego przerwania w rejestrze masek przerw – IMR, wartość wskaźnika stosu - SP i umieścić początek procedury obsługi przerwania w tablicy wektorów przerw – przygotować odpowiedni wektor. Procedura obsługi przerwania jest zatem programem wywołanym pow ustawieniu flagi danego przerwania, który po dokończeniu wykonywanego rozkazu i przy dopuszczeniu przerwania odpowiednimi stanami masek globalnej i indywidualnej zostanie uruchomiony począwszy od własnego wektora w tablicy wektorów przerw. Główne cechy:</p> <ul style="list-style-type: none"> ? Procesor: ? Automatycznie kasuje flagę przerwania, ? automatycznie odsyła na stos jedynie zawartość rejestru PC (czyli adres następnego rozkazu do wykonania po ukończeniu obsługi przerwania), ? automatycznie ustawia globalną maskę przerw INTM co daje zablokowanie przerw. (Zatem inne przerwania nie mogą wystąpić bez zgody programisty), ? tworzy przypisany dla zidentyfikowanego przerwania adres początkowy wektora w tablicy wektorów przerw (pierwszego rozkazu procedury ISR) i umieszcza go w PC, ? odczytuje pierwszy rozkaz spod utworzonego adresu (4-ro słowowy wektor w tablicy zawiera zwykle skok do dalszego ciągu programu procedury ISR), Procedurę ISR charakteryzuje zwykle ? na początku procedury konieczność zachowania (zwykle na stosie) stanu rejestrów procesora używanych w trakcie jej działania i odtworzenie ich zawartości na końcu procedury. <p>43.Dlaczego pojedyncza magistrala zewnętrzna procesora DSP stanowi istotne ograniczenie dla jego szybkości działania? Jeżeli pamięć programu i danych będą umieszczone na zewnątrz procesora, to pojedyncza magistrala zewnętrzna może transportować tylko jeden obiekt. Albo kod rozkazu, albo jedną daną. To zaś uniemożliwia wykorzystanie walorów kolejki i szybkość realizacji programu spada. Może to spowodować obniżenie efektywności o co najmniej 50%</p>	<p>55.W jaki sposób i po co programista może określać/zmieniać położenie tablicy wektorów przerw (początków procedur przerw)? Po resecie sprzętowym procesor nadając wartość rejestrów IPTR równą 0x1FFF będzie sięgał do tablicy wektorów przerw zaczynającej się od adresu 0xFF80.Domyślnie, dla takiej sytuacji tablica wektorów przerw jest lokowana w zakresie adresów od FF80h do FFFFh w przestrzeni pamięci programu.Można przygotować inną/inne tablice w przestrzeni pamięci programu, zaczynające się od adresów równych (IPTR)*128 i wskazać ją procesorowi do użycia poprzez nadanie odpowiedniej wartości rejestrów IPTR a następnie wywołanie programowego reset (czyli rozkazu RESET).</p>
---	--	---	---

