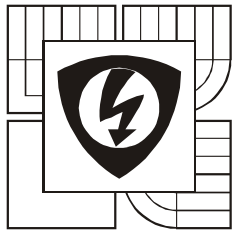
Volací konvence a systém přerušení

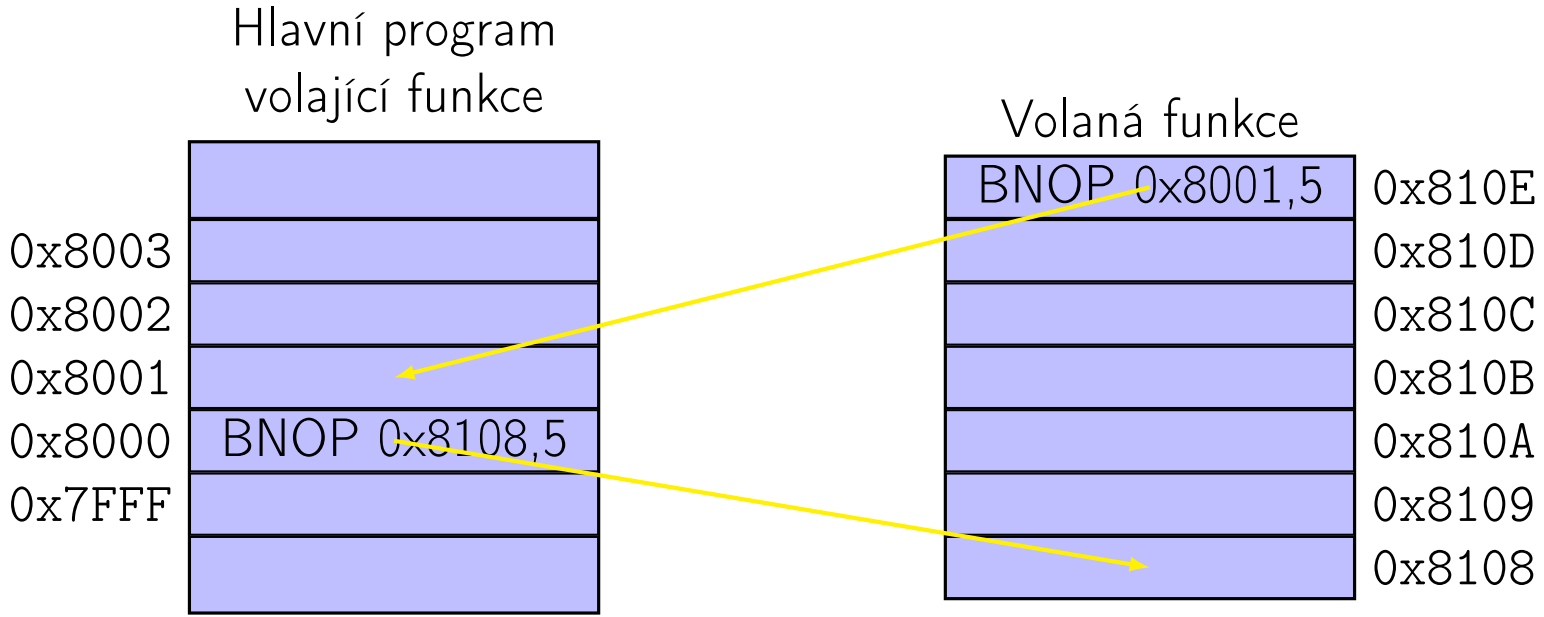


Kurz: MSPR – Signálové procesory

Autor: Petr Sysel

Lektor: Petr Sysel

Příklad volání funkce



Volání funkcí

- Při volání funkcí dodržuje překladač jazyka C volací konvence (calling conventions):
  - adresa vstupního bodu funkce je většinou definována jako znak podtržítko_jméno funkce `_secti` (nebo je možné použít klíčové slovo `asm` před hlavičkou funkce),
  - parametry funkce jsou předávány v definovaných registrech nebo na zásobníku,
  - návratová hodnota je vracena v definovaném registru.
- Při dodržení volacích konvencí je možné volat libovolnou funkci v asembleru z jazyka C a naopak funkci v jazyce C volat z asembleru,
- v případě jazyka C++ s přetěžováním funkcí vstupuje do hry tzv. *name mangling*
  příklad: `class banana{ public: int calories(void);}`⇒`_calories_in_a_banana__Fv`

Volání funkce u překladače Code Composer Studio

- Prvních 10 parametrů funkce je uloženo v registrech `A4`, `B4`, `A6`, `B6`, `A8`, `B8`, `A10`, `B10`, `A12`, `B12`; v případě parametrů více jak 32bitových jsou uloženy v odpovídajících registrových párech;
- zbývající parametry jsou umístěny na zásobník; ukazatel zásobníku je nastaven na první volné místo za parametry;
- pokud je parametrem struktura, je předána adresa struktury v paměti;
- volající funkce si musí uschovat hodnoty registrů `A0` až `A9`, `B0` až `B9`, příp. `A16` až `A31`, `B16` až `B31` pokud chce uchovat jejich hodnotu;
- volající funkce provede skok na vstupní bod volané funkce; návratová adresa je uchována v registru `B3`;
- po návratu musí volající funkce provést uvolnění prostoru na zásobníku.

Použití registrů

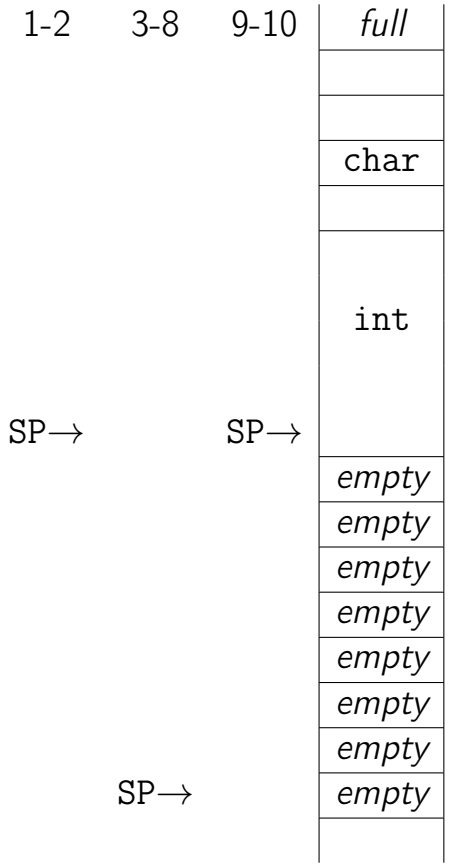
register	zálohuje	použití	register	zálohuje	použití
A0	volající	–	B0	volající	–
A1	volající	–	B1	volající	–
A2	volající	–	B2	volající	–
A3	volající	adresa vracené struktury	B3	volající	návratová adresa
A4	volající	1. argument nebo návratová hodnota	B4	volající	2. argument
A5	volající	1. argument nebo návratová hodnota	B5	volající	2. argument
A6	volající	3. argument	B6	volající	4. argument
A7	volající	3. argument	B7	volající	4. argument
A8	volající	5. argument	B8	volající	6. argument
A9	volající	5. argument	B9	volající	6. argument
A10	volaná	7. argument	B10	volaná	8. argument
A11	volaná	7. argument	B11	volaná	8. argument
A12	volaná	9. argument	B12	volaná	10. argument
A13	volaná	9. argument	B13	volaná	10. argument
A14	volaná	–	B14	volaná	ukazatel haldy (DP)
A15	ukazatel dat (FP)	–	B15	volaná	ukazatel zásobníku (SP)
A16–A31	volající	pouze u TMS320C6400	B16-B31	volající	pouze u TMS320C6400

Použití zásobníku

- Ukazatel zásobníku B15 (alias SP) musí vždy ukazovat na vrchol zásobníku – první prázdná pozice,
- ukazatel zásobníku musí být zarovnán na násobky 8 (největší současně přenositelná velikost dat 64 bitů),
- vytvoření místa na zásobníku je možné dekrementací ukazatele zásobníku,
- před návratem z funkce je nutné obnovit ukazatel zásobníku na původní pozici.

Příklad využití zásobníku

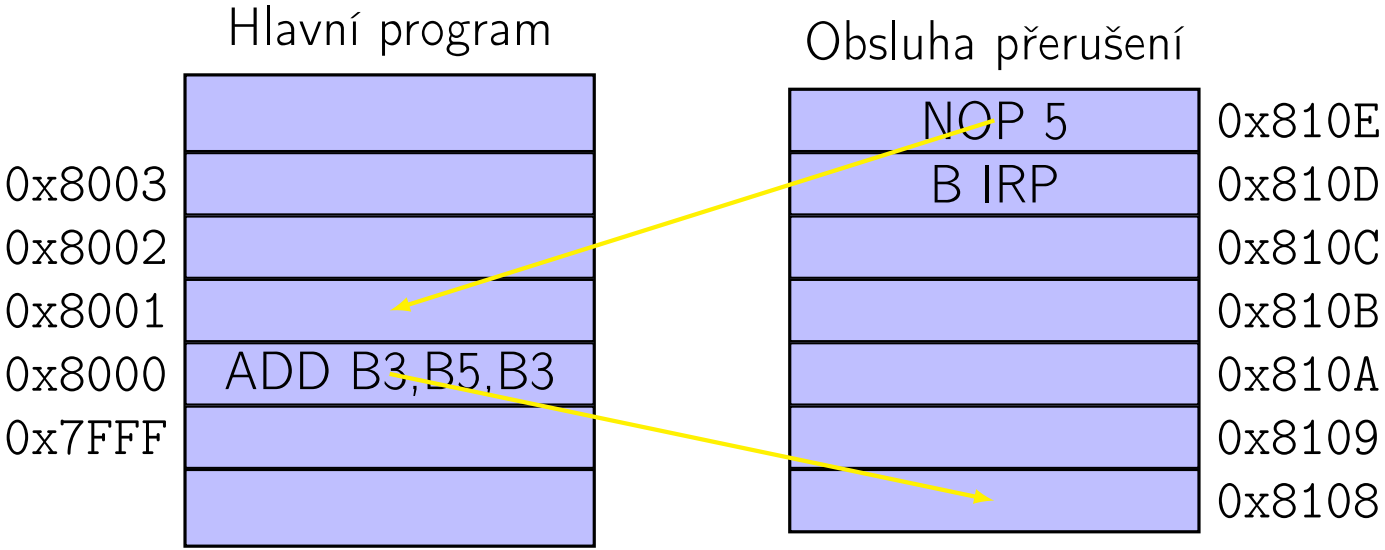
```
1  _func:
2      STW     B3,*SP--[2]; vytvoření místa
3
4      LDW     *+SP[2],A6 ; přístup k int
5
6      LDB     *+SP[13],B5 ; přístup k char
7
8      LDW     *++SP[2],B3; uvolnění místa
9
10     NOP     5
11
12     B       B3          ; návrat z funkce
```



Přerušení programu – základní pojmy

- Procesor je schopen komunikovat s okolím standardními instrukcemi pro zápis nebo čtení z periferních obvodů,
- okamžik čtení nebo zápisu tak určuje procesor a většinou se jedná o asynchronní obsluhu,
- často je nutné reagovat na vnější událost okamžitě – synchronně – hned jak nastane:
  - nebezpečné zvýšení teploty, otáček, ...,
  - připojení nadřazeného počítače,
  - stisk tlačítka STOP,
- přerušením je vnějším zařízením vyžádána okamžitá obsluha,
- úlohou řadiče přerušení je:
  - registrovat aktivní požadavky na přerušení,
  - v případě více než jednoho aktivního požadavku je řadit podle priority,
  - identifikovat zdroj přerušení, který byl vybrán pro obsloužení,
  - informovat procesor o vzniku žádosti o přerušení.

Příklad obsluhy přerušení



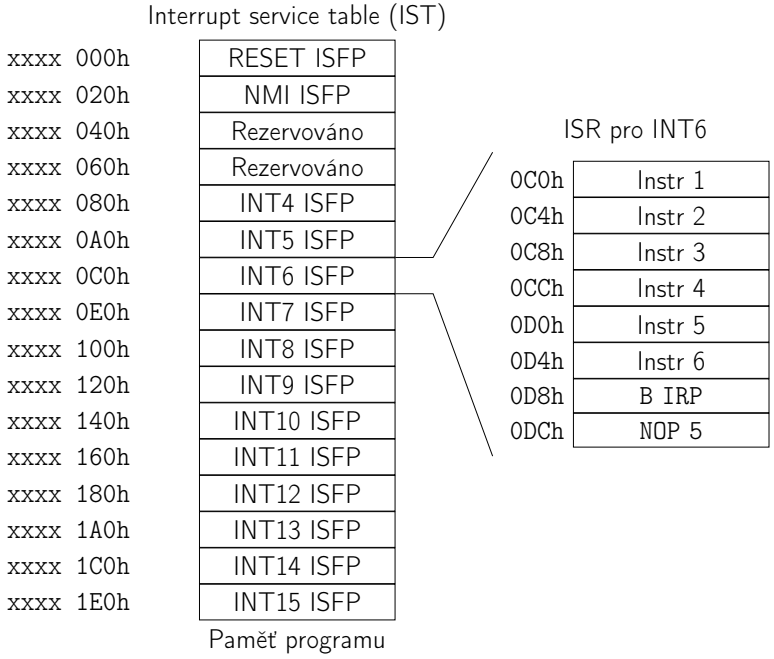
Rozdíly mezi funkcí a přerušením

- Přerušení může vzniknout kdykoliv – musí být transparentní,
- po návratu musí být procesor ve stejném stavu jako před začátkem obsluhy přerušení,
- musí být zálohovány registry,
- musí být zálohován stavový registr,
- návrat z přerušení se provede speciální instrukcí, která obnoví stavový registr,
- v jazyce C to zajistí klíčové slovo `interrupt` u deklarace funkce.

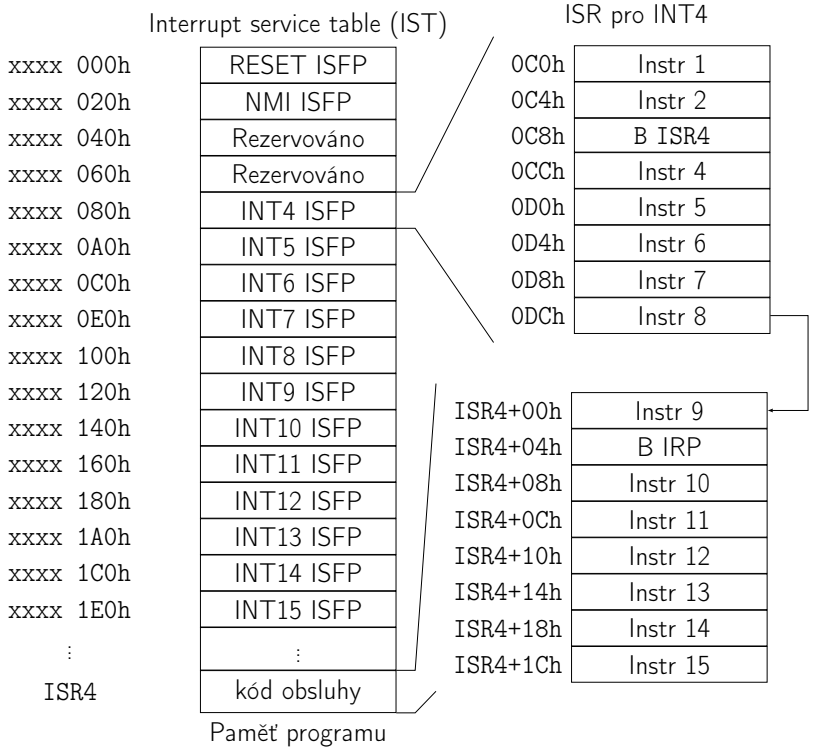
Tabulka vektorů přerušení

- Pro každý zdroj přerušení musí být jednoznačně určena adresa počátku obslužného podprogramu,
- většinou jsou obslužné podprogramy (vektory přerušení) sestaveny do tabulky vektorů přerušení,
- vektor přerušení se může skládat z jediné instrukce skoku do podprogramu nebo z několika málo instrukcí (u TMS320C6416 z 8 instrukcí – jednoho instručkního paketu),
- návrat z přerušení je nutné provádět speciální instrukcí, která zajistí obnovení stavového registru (u procesoru TMS320C6000 instrukce `B IRP`),
- protože přerušení může být přerušeno nemaskovatelným přerušením, návrat z něho je nutné provést instrukcí `B NRP`,
- adresa počátku tabulky vektorů přerušení je uložena ve zvláštním registru (u procesoru TMS320C6416 registr `ISTP`).

Obsluha krátkého přerušení



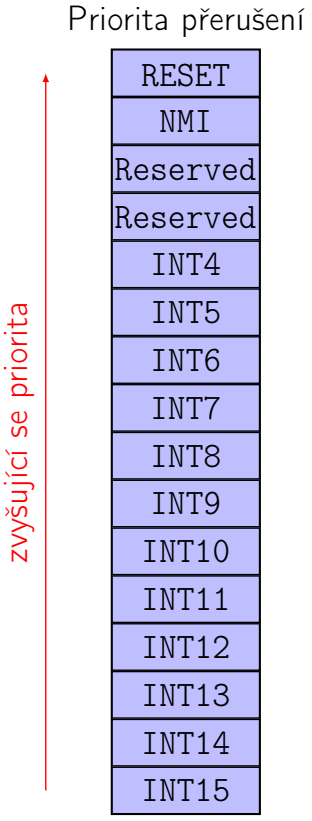
Obsluha dlouhého přerušení



Priorita přerušení

- V případě více zdrojů přerušení je nutné rozhodnout, které přerušení bude obslouženo,
- každé přerušení má přiřazenu prioritu – přerušení s nižší prioritou může být přerušeno jen přerušením s vyšší prioritou,
- přidělení priority může být:
  - pevná priorita – každý signál přerušení má pevně danou prioritu, většinou signál `IRQ0` má nejvyšší, `IRQ1` nižší, ..., `IRQ`N nejnižší,
  - cyklická priorita – při každé obsluze přerušení se nejvyšší priorita posune vpravo – nejvyšší prioritu bude mít `IRQ1`, po dalším přerušení `IRQ2`, ...,
  - volitelná priorita – každému signálu přerušení lze přidělit vhodnou prioritu v konfiguračním registru řadiče přerušení – v nejjednodušší podobě se přepíná mezi nízkou a vysokou prioritou,
- většina procesorů podporuje jeden zvláštní signál pro obsluhu přerušení s nejvyšší prioritou, který navíc nelze maskovat – nemaskovatelné přerušení `NMI`

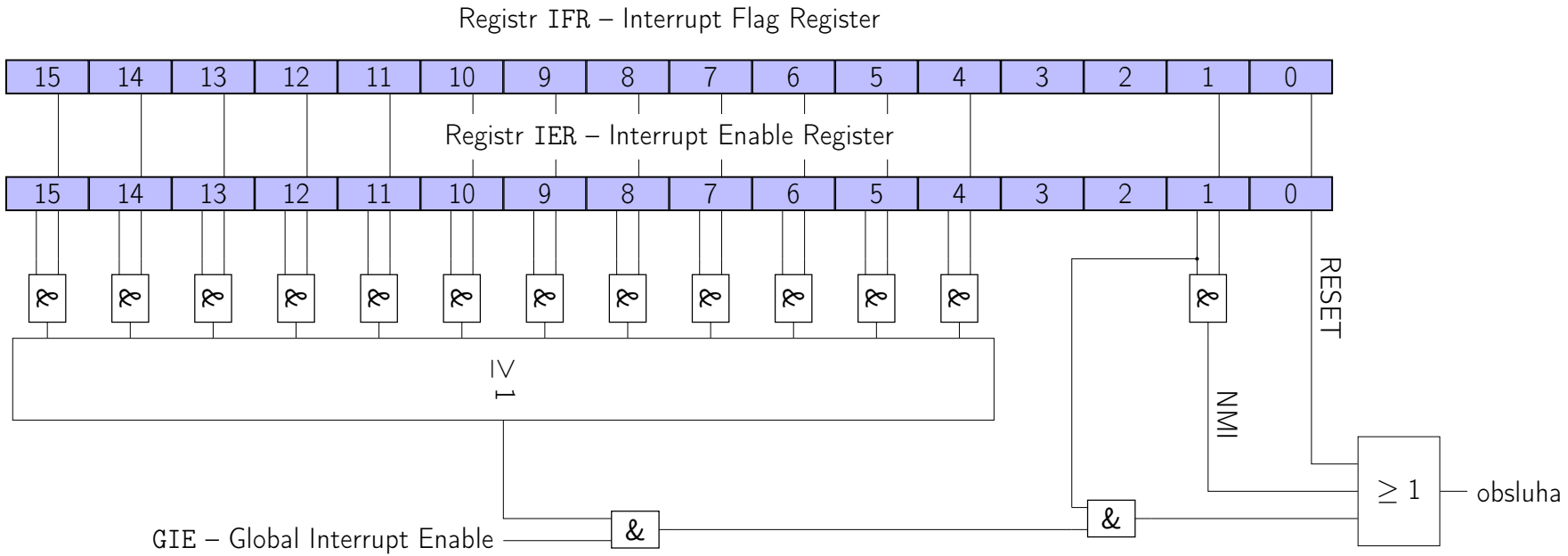
Priorita přerušení u TMS320C6416



Maskování přerušení

- Ne všechny požadavky na přerušení musí být obslouženy – např. pokud některou periferii nepoužívám, je zrovna obsluhována přerušení z vyšší prioritou, . . . ,
- jedním bitem konfiguračních registrů řadiče přerušení lze zakázat veškerá přerušení – standardně bývá přerušení zakázáno po resetu,
- lze zakázat i jednotlivá přerušení samostatně – maskovat přerušení:
  - v jednom registru řadiče přerušení jsou registrovány požadavky na přerušení (interrupt pending/request register),
  - v druhém registru jsou jednotlivá přerušení povolena (interrupt mask register),
  - přerušení je dále zpracováno jen pokud je vyvolán požadavek na přerušení a dané přerušení je povoleno,
- priorita právě obsluhovaného přerušení je zapsána do registru a dále jsou zpracovávány jen přerušení s vyšší prioritou – se stejnou a nižší prioritou jsou maskovány.

Maskování přerušení u TMS320C6416



Maskování přerušení u TMS320C6416

- Registr `IFR` – Interrupt Flag Register – registruje příchozí žádosti o přerušení,
- registr `IER` – Interrupt Enable Register – povoluje nebo zakazuje obsluhu daného přerušení,
- přerušení od `RESET` je povoleno vždy (nejnižší bit `IER` je vždy 1),
- nastavením bitu 1 povolíme nemaskovatelné přerušení `NMI` – Non Maskable Interrupt,
- nemaskovatelné přerušení již nelze softwarově zakázat – bit 1 je nulován pouze při obsluze nemaskovatelného přerušení nebo po resetu,
- ostatní přerušení je nutné povolit bitem `GIE` v registru `CSR`,
- pokud je obsluhováno nemaskovatelné přerušení, jsou všechna ostatní také zakázána.

Postup při obsluze nemaskovatelného přerušení

- Zahájení obsluhy:
  - Bit 1 `NMIE` registru `IER` je vynulován, tzn. další nemaskovatelná přerušení jsou maskována,
  - rozpracované instrukce jsou anulovány,
  - do registru `NRP` – Non maskable Return Pointer – je uložena návratová adresa,
  - je proveden skok na vektor nemaskovatelného přerušení.
  - vynuluje se bit 1 registru `IFR` žádosti o nemaskovatelné přerušení.
- Návrat z obsluhy:
  - Instrukce `B NRP` naplní `PC` hodnotou návratové adresy z `NRP`,
  - po pěti zpožďovacích cyklech je provedena instrukce z návratové adresy,
  - provede nastavení bitu 1 registru `IER` – nemaskovatelná přerušení jsou opět povolena.

Postup při obsluze maskovatelného přerušení

- Zahájení obsluhy:
  - Bit 0 `GIE` (Global Interrupt Enable) registru `CSR` je zkopírován do bitu 1 `PGIE` (Previous GIE),
  - bit 0 `GIE` je vynulován – všechna další přerušení jsou zakázána,
  - rozpracované instrukce jsou anulovány,
  - do registru `IRP` – Interrupt Return Pointer – je uložena návratová adresa,
  - je proveden skok na příslušný vektor přerušení,
  - vynuluje se příslušný bit registru `IFR` žádosti o přerušení.
- Návrat z obsluhy:
  - Instrukce `B IRP` naplní `PC` hodnotou návratové adresy z `IRP`,
  - po pěti zpožďovacích cyklech je provedena instrukce z návratové adresy,
  - obnoví hodnotu bitu `GIE` z bitu `PGIE` – přerušení jsou opět povolena.

Základy práce s přerušením

- Po resetuje přerušení vždy zakázáno,
- před povolením přerušení je nutné:
  - nastavit ukazatel zásobníku SP,
  - nastavit adresu začátku tabulky vektorů přerušení,
  - nastavit vektory přerušení pro požadované přerušení,
  - přiřadit prioritu pro požadované přerušení,
  - nastavit maskování přerušení,
  - povolit přerušení,
- při obsluze uložit hodnoty měněných registrů tak, aby mohly být obnoveny,
- před koncem přerušení je nutné příznak přerušení vynulovat,
- nepoužité vektory přerušení je vhodné nastavit, aby byly obslouženy pomocí diagnostických podprogramů.

Základy práce s přerušením

- Přerušení je možné vyvolat softwarově nastavením příslušného bitu registru `IFR`,
- nastavení je provedeno zápisem do registru `ISR` – Interrupt Set Register,
- opačně zápisem do registru `ICR` – Interrupt Clear Register – je možné příznak žádosti o přerušení v registru `IFR` vynulovat,
- pokud je vyžadováno vnořené přerušení, je nutné:
  - zálohovat registry `CSR` a `IRP` (resp. `NRP` v případě nemaskovatelného přerušení) na zásobník,
  - zálohovat registr `IER` na zásobník a zakázat přerušení s nižší prioritou (není nutné, ale vhodné),
  - povolit přerušení nastavením bitu `GIE` (resp. `NMIE` v případě nemaskovatelného přerušení),
  - před návratem z přerušení je nutné registry opět obnovit.