Obecný popis křížových překladačů (kompilátorů) pro mikroprocesory řady 8x51/52

Doc.Ing.Rudolf Jalovecký,CSc. (vyvoj@jalsoft.iol.cz, Rudolf.Jalovecky@vabo.cz)

1. Úvod

V assemblerech pro mikroprocesory řady 8x51/52, ale i ADuC812/16/24 se setkáváme s pojmy „direktivy překladače- kompilátoru“. Tyto direktivy bezprostředně řídí vlastní překlad překladače. Předložený text má za cíle obecné seznámení a použití těchto direktiv bez ohledu na konkrétní typ překladače i když je zaměřen především na direktivy těch typů kompilátorů, které překládají zdrojový text přímo do absolutního kódu (*.HEX), nikoliv o tvaru objektivního překladu (*.OBJ), který následně musí být ještě spojen - slinkován s knihovnou procedur a funkcí.

2. Obecný popis kompilátorů

Kompilátor je křížový překladač jazyka symbolických adres jednočipového mikropočítače rodiny 8x51/52. Jeho vstupem pro kompilátor je zdrojový soubor <jméno>.ASM v jazyce symbolických adres, výstupem pak především výsledný (cílový) soubor ve formátu INTEL-HEX <jméno>.HEX a volitelně například protokol o překladu <jméno>.PRN nebo <jméno>.LST Vzhledem k velkému množství variant kompilátorů, které jsou k dispozici, je nutné předem konstatovat, že nejsou ve velké většině 100% kompatibilní.

Překladače se zpravidla spouští z příkazového řádku operačního systému nebo v dávce, použitím příkazu s několika parametry, např.:

ASM51 <jméno>[.ASM] [/<parametry řídící překlad>]

kde <jméno> je jméno zdrojového programu, má-li příponu .ASM není potřebné ji uvádět,

<parametry řídící překlad> - např. parametr pro generování tabulky symbolů nebo parametr pro vytvoření tabulky křížových odkazů apod.

3. Popis syntaxe

Přesný sémantický význam jednotlivých instrukcí je popsán v dostupné literatuře a není nutné jej zde uvádět. Uveďme jen použití direktiv kompilátorů, které mají bezprostřední význam na přeložený program.

3.1 Rezervovaná jména

Rezervovaná jména mají v překladači vyhrazeny konkrétní význam a programátor je může použít pouze předem určeným způsobem.

Seznam rezervovaných jmen:

A	AB	AC	ACALL	ACC	ADD	ADDC	AJMP
AND	ANL	AR0	AR1	AR2	AR3	AR4	AR5
AR6	AR7	B	BIT	C	CALL	CJNE	CLR
CPL	CY	DA	DATA	DB	DEC	DIV	DJNZ
DPH	DPL	DPTR	DSTR	DW	EA	ELSE	END
ENDIF	EQU	ES	ET0	ET1	EX0	EX1	F0
HIGH	IE	IE0	IE1	IF	INC	INT0	INT1
IP	IT0	IT1	JB	JBC	JC	JMP	JNB
JNC	JNZ	JZ	LCALL	LJMP	LOW	MOV	MOVC
MOVX	MUL	NOP	NOT	OR	ORG	ORL	OV
P	P0	P1	P2	P3	PC	PCON	POP
PS	PSW	PT0	PT1	PUSH	PX0	PX1	R0
R1	R2	R3	R4	R5	R6	R7	RB8
RD	REN	RET	RETI	RI	RL	RLC	RR
RRC	RS0	RS1	RXD	SBUF	SCON	SET	SETB
SHL	SHR	SJMP	SM0	SM1	SM2	SP	SUBB
SWAP	T0	T1	TB8	TCON	TF0	TF1	TH0
TH1	TI	TL0	TL1	TMOD	TR0	TR1	TXD
WR	XCH	XCHD	XRL				

3.2 Jména definovaná programátorem

Jména definovaná programátorem mohou být tvořeny písmeny, číslicemi a znakem podtržení '_'. První znak musí být vždy pouze písmeno. Maximální délka jména může být 80 znaků, překladač však pracuje pouze s prvními 8 znaky, které také rozlišuje.

3.3 Konstanty

Překladač rozlišuje 5 možných způsobů zápisu konstant.

Číselné konstanty :

a) dekadické - číslice 0-9, mohou končit znakem D,

b) binární - číslice 0 nebo 1 končící povinně znakem B,

c) hexadecimální - číslice 0-9 znaky A-F, definice musí začínat číslicí a musí končit znakem H,

d) oktalové - číslice 0-7 končící znakem O nebo Q.

Znakové konstanty :

e) ASCII znak - libovolný ASCII znak, který je uzavřený do jednoduchých uvozovek.

Zvláštní význam ve výrazu má znak $ - dolar, kterému je při překladu přiřazena aktuální hodnota programového čítače.

3.4 Operátory

Ve výrazech mohou být jména a konstanty spojena operátory. Kompilátory rozpoznávají následující operátory :

operátor	**operace**	**operátor**	**operace**
+	sečítání	SHL	bitový posun vlevo
-	odečítání	SHR	bitový posun vpravo
/	dělení	NOT	negace
*	násobení	LOW()	dolní byt operandu
OR	logický součet	HIGH()	horní byt operandu
AND	logický součin		

Vyhodnocování operátorů při jejich kombinaci probíhá podle priority. Následující seznam je uspořádán podle vzrůstající priority operátorů + - / * *AND OR SHL SHR NOT HIGH LOW*. Při stejné úrovni priority je výraz vyhodnocován zleva doprava. Pořadí vyhodnocování může být změněno použitím závorek.

4. Formát instrukcí

Snad všechny kompilátory pracují s následujícím formátem instrukce:

[návěští:] (pseudo)instrukce [operand(y)] [;komentář]

Délka jednotlivých polí není určena. Pro vyšší přehlednost a pohodlnější práci se však doporučuje vyhradit si pro každé pole 8 znaků. Většina editorů má tabulátor nastaven standardně na tuto velikost, komentář doporučuji zapisovat až po dvou tabulátorech..

4.1 Návěští

Návěští je symbolické jméno, které je ukončeno dvojtečkou. Tomuto jménu je při překladu přiřazena konkrétní adresa, na kterou je možné se přes identifikátor návěští odvolávat. Dané návěští smí být deklarováno pouze jednou. Při vícenásobném výskytu téhož jména v poli návěští hlásí překladač chybu. Rozhodujících je opět prvních 8 znaků.

4.2 Instrukce, pseudoinstrukce

Většina kompilátorů umožňuje používat některé z následujících pseudoinstrukcí:

Definice uživatelských jmen - **EQU, SET, DATA**

Syntaxe: *<jméno> EQU <výraz>*

<jméno> SET <výraz>

<jméno> DATA <výraz>

Symbolickému jménu je přiřazena hodnota získaná vyhodnocením výrazu. Rozdíl mezi pseudoinstrukcí EQU a SET je ten, že použitím SET může být jednomu jménu přiřazena různá hodnota vícekrát (jedná se tedy o přiřazení hodnoty), kdežto použití EQU odpovídá definici a toto jméno už nesmí být definováno na jiném místě.

Pseudoinstrukce DATA je určena k definici symbolických jmen, které se odkazují na adresy vnitřní paměti dat (direct address).

Pseudoinstrukcí EQU je možné rovněž přiřadit symbolické jméno některému z registrů R0 až R7 procesoru. Je tedy možný zápis jako např.:

BASE EQU R0

Poznámka: Symbolická jména registrů musí být definována před jejich prvním použitím.

Definice uživatelských označení bitů - **BIT**

Syntaxe: *<jméno> BIT <hodnota>*

Tato pseudoinstrukce umožňuje přiřadit přímo adresovatelným bitům ve vnitřní paměti dat symbolická jména. Rovněž umožňuje přiřadit těmto jménům i I/O porty. Jako hodnota může být použita konstanta z intervalu 0-FFH nebo symbolické označení bitu dle mnemoniky výrobce.

Například:

```
PRVNI    BIT   34H
VYSTUP   BIT   P3.2
TRETI    BIT   ACC.3
```

Nastavení programového čítače - **ORG**

Syntaxe: *[návěští:] ORG <výraz>*

Tato pseudoinstrukce provede nastavení hodnoty programového čítače na hodnotu danou vyhodnocením výrazu. Při pokusu o nastavení větší hodnoty než je rozsah adres (0 až FFFFH), ohlásí překladač chybu.

Definice dat - **DB, DW**

Syntaxe: *[návěští:] DB <výraz> [,<výraz>]*

Pseudoinstrukce postupně ukládá jednobytové hodnoty, získané vyhodnocením jednotlivých výrazů, na aktuální adresy, počínaje současnou adresou programového čítače. Výrazy musí nabývat hodnot v rozsahu 0 - FFH. Povoleny jsou také řetězce znaků uzavřené v jedno-

duchých uvozovkách, např. posloupnost znaků '123' se uloží jako 31H,32H,33H. Počet výrazů za jedním příkazem DB je omezen pouze délkou řádku.

Syntaxe: *[návěští:] DW <výraz> [,<výraz>]*

Pseudoinstrukce postupně ukládá dvoubytové hodnoty, získané vyhodnocením jednotlivých výrazů, na aktuální adresy, počínaje současnou adresou programového čítače. Výrazy musí nabývat hodnot v rozsahu 0 až FFFFH. Počet výrazů za jedním příkazem DW je omezen pouze délkou řádku.

POZOR! Vyšší byte je ukládán na nižší adresu a nižší byte na vyšší adresu.

Rezervování paměti - **DS**

Syntaxe: *[návěští:] DS <výraz>*

Při překladu je v paměti rezervován počet bytů daný výrazem. Pro překladače, které přímo generují HEXa kód se tato pseudoinstrukce příliš nepoužívá, neboť již není možné rezervované místo využít.

Poznámka: V tomto případě musí být hodnoty případných proměnných, použitých ve výrazu, definovány před jeho vyčíslením.

Výběr použité banky registrů - **USING**

Syntaxe: *[návěští:] USING <výraz>*

Pseudoinstrukce definuje použití příslušné banky registrů od místa svého výskytu. Výraz po vyhodnocení musí nabývat hodnot z intervalu 0 až 3. Při zpracování instrukcí, které pracují s registry R0 až R7, je při překladu zohledněna zvolená banka registrů tehdy, je-li na místě registrů v instrukci uveden operand ARx (Absolute Register), kde x je číslo registru 0 až 7. V takovém případě přeloží překladač instrukci jakoby byla použita instrukce s přímou adresou. Implicitně je zvolena banka 0.

Z praktických zkušeností nedoporučuji tuto direktivu příliš používat, je vhodné raději zvolit přímý assemblerovký příkaz pro nastavení banky registrů.

Podmíněný překlad - **IF, ELSE, ENDIF**

Syntaxe: *[návěští:] IF <výraz> [návěští:] ELSE [návěští:] ENDIF*

Překladač vyhodnotí výraz a pokud je výsledek různý/roven od nuly, přeloží/nepřeloží následující sekvenci instrukcí až po výskyt pseudoinstrukce ELSE nebo ENDIF. Při výskytu pseudoinstrukce ENDIF je podmíněný překlad ukončen. Při výskytu ELSE se podmínka obrátí a další instrukce až po ENDIF se nepřekládají/překládají.

Poznámka: Vnoření pseudoinstrukcí IF, ELSE, ENDIF nemusí být u některých kompilátorů povoleno.

Ukončení zdrojového textu - **END**

Syntaxe: *[návěští:] END*

Nepovinná pseudoinstrukce, která oznamuje překladači konec překládané části zdrojového textu. Jakýkoli další text je ignorován, je tedy možné si za tento příkaz „uložit“ část ještě nezpracovaného, neodladěného programu Chybí-li tato pseudoinstrukce na konci zdrojového textu, je v protokolu o překladu uvedena varovná zpráva.

4.3 Operandy

Pokud je na místě operandu přímá (direct) adresa, je možno použít buď numerickou hodnotu (z intervalu 0-FFH) nebo symbolické jméno speciálního funkčního registru. Stejné pravidlo platí i pro bitový operand (bit) s tím, že jako symbolických jmen se používá názvů přímo adresovatelných bitů podle zavedené mnemotechniky.

Pokud se na místě přímého nebo bitového operandu vyskytne hodnota, která není definovaná, překladač na tuto skutečnost zareaguje vypsáním varovné zprávy do protokolu o překladu a výsledek takovéto operace není definovaný.

Je-li jako operand uvedena přímá hodnota (#data), je možno použít výrazu, jehož vyčíslením se tato hodnota získá. Ve výrazu se mohou vyskytovat čísla a symbolická jména.

Příklad:
```
LJMP $+1000H
MOV A,#12
MOV A,#1+2*CISLO
CISLO EQU 101B
```

4.4 Komentář

Komentář je libovolný text, který začíná středníkem. Narazí-li překladač při překladu na středník, pak celý zbytek textu do konce řádky považuje za poznámku. Mimo obecný formát instrukce jsou možné i tyto formáty:

[návěští:] ;komentář ;komentář

Obecně je doporučeno používat velké množství poznámek a komentářů neboť časem alespoň lépe poznáte svoji původní zprogramovanou myšlenku.

5. Protokol o překladu

Snad všechny překladače vytváří (mimo vlastního přeloženého HEXa souboru) i soubor o provedeném překladu, tzv. protokol (*.PRN, *.LST). Tento protokol o překladu zahrnuje opis zdrojového textu, přeložený cílový kód, případné chybové hlášení a někdy i volitelně tabulku symbolů a křížových odkazů.

Obecný formát řádku protokolu o překladu:

<adresa> <cíl.kód> <opis zdrojového textu>

kde - adresa je tvořena čtyřmi číslicemi v hexadecimálním tvaru a udává absolutní adresu, na kterou se ukládá cílový kód

- cílový kód je tvořen hexadecimálními číslicemi operačního kódu a případných operandů
- opis zdrojového textu je odpovídající řádek zdrojového textu včetně komentáře.

5.1 Chybová hlášení při překladu

Skončí-li překlad bez chyby, pak je na konci protokolu o překladu uveden text :

No errors detected

Je-li při překladu zjištěna chyba, je protokol ukončen textem :

<x> errors detected

kde x je počet zjištěných chyb.

5.1.1 Chyby programu

Při výskytu chyby v překládaném programu generuje překladač do protokolu o překladu chybové hlášení ve tvaru:

**** Error *** <typ chyby>*

Toto chybové hlášení je vždy uvedeno pod řádkem, ve kterém se chyba vyskytla.

Přehled standardních chybových hlášení:

identifier not declared	identifikátor není deklarován
val.out of range 0..FFH	hodnota přesahuje rozsah 1 bytu
out of range 0..FFFFH	hodnota mimo rozsah 1 slova
out of current page	skok mimo současnou stránku
out of address range	odkaz na adresu mimo adresový rozsah
multiply declaration	vícenásobná deklarace
syntax error	syntaktická chyba
not assigned to reg.	identifikátor není přiřazen registru R0,R1
register expected	očekávají se pouze registry R0,R1
multiply assignment	vícenásobné přiřazení
multiply register assigned	registr je přiřazen dvakrát
multiply label declared	násobná deklarace návěští PC
out of range	programový čítač je mimo rozsah
nested IF	vnořená pseudoinstrukce IF
nested ELSE	vnořená pseudoinstrukce ELSE
nested ENDIF	vnořená pseudoinstrukce ENDIF
assigned to register	na místě pro direct adresu jen použita proměnná, přiřazená reg.

out of +127 -128 range	relativní skok je mimo daný rozsah
out of 0 - 3 range	za pseudoinstrukcí using použita hodnota větší než 4
<symbol> expected	je očekáván <symbol>

5.1.2 Upozornění PŘEKLADAČE

Při překladu mohou vzniknout situace, které nejsou chybou v překládaném programu, ale programátor by měl být o tom informován. Toto chybové hlášení je vždy uvedeno pod řádkem, ve kterém se situace vyskytla.

Warning - bad SFR address	chybná adresa speciálního funkčního registru
Warning - bad bit address	chybná adresa bitově adresovatelné paměti
Warning - END expected	očekáván konec souboru

Tato varování mají pouze charakter informace, nejsou považovány za chybu.

5.1.3 Chyby systému

Pokud dojde při překladu k chybě systému, generuje překladač do protokolu o překladu chybové hlášení ve tvaru:

*** System error *** <typ chyby>

Seznam chybových hlášení:

Out of memory	- program má nedostatek paměti pro ukládání dat
No such file	- pokus o otevření neexistujícího souboru
Disk is full	- plný disk
Can't open file	- systém nepovolí otevřít další soubor. Je třeba zkontrolovat velikost proměnné FILES v souboru CONFIG.SYS

5.2 Tabulka symbolů

Je-li při spuštění překladače zadán požadavek na vytvoření tabulky symbolů, je vytvořena tabulka symbolů a připojena na konec protokolu o překladu.

5.3 Tabulka křížových odkazů

Je-li při spuštění překladače zadán požadavek na vytvoření tabulky křížových odkazů je vytvořena tabulka křížových odkazů a připojena na konec protokolu o překladu. V tabulce jsou uvedeny jednotlivá symbolická jména definovaná programátorem a pro každý výskyt jména číslo řádku, ve kterém je jméno použito. Číslo řádku, ve které je identifikátor definován, je uvozeno znakem #.

6. Poznámky na závěr

Předložený text by měl sloužit k prvotnímu seznámení se s možnostmi jak ovlivnit překlad zdrojového textu v kompilátoru. Současně však také může dávat návod na obecnější využití identifikátorů paměťových míst, konstant a konkrétních adres I/O portů.

Je vhodné také připomenout na skutečnost, že ne vždy najdeme standardní označování proměnných v jednom a téže kompilátoru. Příkladem tomu může být označování akumulátoru A, který někdy musíme zapsat jako Acc (PUSH Acc, ale MOV A,...). Dále je vhodné upozornit na skutečnost, že většina překladačů nerozlišuje velké a malé znaky, je tedy jedno zda zapíšete mov a,b nebo MOV A,B. Obecně se však doporučuje, obdobně jako strukturování zápisu, používat zápis instrukcí ve velkých písmenech.

7. Obsah

8. Použité prameny

- Popis křížového překladače EASY CASS51
- Programovací jazyk assembler 8051, TESLA ELTOS,1987
- 8051 CROSS ASSEMBLER, MetaLink Corporation
- Internetovské stránky firem, poskytující kompilátory pro mikroprocesor 8x51/52
- Osobní poznámky a zkušenosti