SDL-processzor

SZEGHY ISTVÁN
BHG Fejlesztési Intézet

ÖSSZEFOGLALÁS

A cikk az SDL grafikus tervezési módszer (CCITT) rövid bemutatása után, annak egy szöveges leképzési rendszerét ismerteti, amiből — alkalmas fordító programmal — olyan kódsorozatot lehet generálni, ami közvetlenül az I 8085 mikroprocesszoron futtatható. Így a hierarchikus, absztrakt SDL-processzorokat alkalmazó modell nemcsak áttekinthetővé teszi a rendszertervezést, hanem egyszerűen realizálhatóvá is.

SZEGHY ISTVÁN

Egyetemi tanulmányait a Budapesti Műszaki Egyetemen végezte, ahol 1954-ben gyengeáramú villamosmérnöki oklevelet kapott. 1954—1958 között üzemmérnök az Elektronikus Mérőkészülékek Gyárában (EMG). 1958-tól az Elektromechanikai Vállalatnál (EMV) mint fejlesztőmérnök, majd laborvezető, kommunikációs rádióadók fejlesztésével foglalkozott. Jelenleg a BHG Fejlesztési Intézetében (az EMV jogutódjánál) annak a laboratóriumnak a vezetője, ahol egy digitális telefon alközpont család software fejlesztése folyik.

1. Bevezetés

A véges automaták működésének leírására, illetve tervezésére igen hatékony módszer a belső állapotok és azok közötti átmenetek definiálása.

Ezt támogatja a CCITT-ben kidolgozott és általa ajánlott Functional Specification and Description Language (SDL) [1] néven ismert grafikus ábrázolási rendszer, amiben néhány egyszerű szimbólum segítségével pontos és jól áttekinthető kép rajzolható egy automata működéséről.

Az SDL ábra felfogható úgy, mint egy program, amit az automatát vezérlő processzornak kell végrehajtania. Ezt a programot azonban olyan kódsorozattá kell transzformálni (lefordítani), amit a valós processzor képes értelmezni és végrehajtani.

Az ilyen, SDL ábra alapján történő kódgenerálásra az irodalomban több módszer is ismeretes [2], [3].

Ebben a cikkben egy olyan további módszert ismertetünk az SDL ábra alapján történő realizálásra, ami az SDL képet egyfelől absztrakt processzorként kezeli, másfelől úgy fogja fel, mint egy programot, amiből a valós processzor számára gépi kódot generál.

A módszer a DIPEX digitális telefon alközpont [4] fejlesztése során került kidolgozásra, mint a software fejlesztés eszköze. Ezért felhasználásának bemutatására is erről a területről választottunk példákat.

2. Az SDL leírás

Az alábbiakban röviden összefoglaljuk az SDL alapjait, melyek a szóban forgó módszer szempontjából fontosak. További részletek az irodalomban hozzáférhetőek.

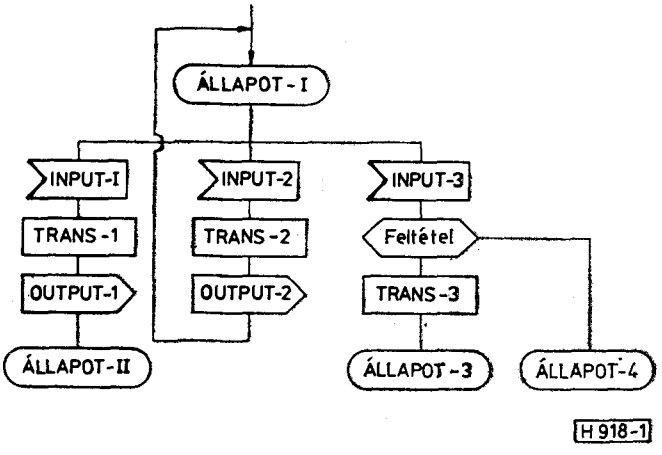
Az SDL a véges automaták működését a belső állapotok egymást követő láncával írja le.

Az automata egy belső állapotból valamilyen beérkező jel hatására meghatározott tranzakcióval (műveletvégzéssel) megy át egy másik belső állapotba, vagy tér vissza eredeti állapotába.

Valójában ezek a tranzakciók jelentik az automata működését, amit a külső szemlélő is érzékelni tud.

Ebben a felfogásban elegendő néhány szimbólumot definiálni ahhoz, hogy a véges automata működési rendszere leírható legyen. Az SDL-ben ezek a következők:

- állapot,
- bemenet,
- kimenet,
- tranzakció,
- döntés.



1. ábra. Egy lehetséges SDL-kép bemutatása

Az általános szimbólumok a megrajzolt ábrában megnevezéssel szerepelnek, és ez a megnevezés eredetükre, hatásukra, vagy műveleti, illetve döntési sajátosságaikra utal.

Az 1. ábra bemutat egy lehetséges SDL leírást, amit szavakkal az alábbiak szerint értelmezhetünk:

Beérkezett: 1983. XI. 18. (#)

Kezdetben az automata az ÁLLAPOT-1 névvel jelölt állapotban van. Ebben az állapotában mindaddig megmarad, amíg az INPUT-1, INPUT-2 vagy INPUT-3 megnevezésű bemenő jelzések valamelyike be nem érkezik. Ezután a működés a következő:

1. Ha INPUT-1 jelzés érkezett, a TRANS-1 művelet hajtódik végre. Ennek hatására generálódik az OUTPUT-1 kimenő jelzés, majd ezek után az automata átkerül az ÁLLAPOT-2 nevű állapotba.
2. Az INPUT-2 jel beérkezése hatására a TRANS-2 műveletnek kell végrehajtódnia, amit az OUTPUT-2 jelzésátadás követ, és az automata marad eredeti állapotában.
3. Ha INPUT-3 jel érkezett be, akkor egy döntési folyamat indul, aminek eredményétől függően vagy közvetlen állapotváltozás történik, vagy előbb végrehajtódik a TRANS-3 művelet, majd az automata felveszi az ÁLLAPOT-3 állapotot.

3. Az SDL-processzor

Ha az SDL ábrát absztrakt processzornak tekintjük, akkor annak működéséről a következőket mondhatjuk:

A processzor mindaddig várakozó állapotban marad, amíg valamelyik bemenő jelzés be nem érkezik azok közül, amiket az SDL leírás ehhez az állapothoz bemenetként hozzárendelt.

Ez a jelzés a megfelelő input periférián aktivizálja a processzort, ami a memóriájában tárolt állapotinformáció és az aktuális input jel alapján azt a műveletsorozatot hajtja végre, amit az SDL leírás előír. Ha eközben állapotszimbólumot talál, ennek az állapotnak a jellemzőit elmenti a memóriájába és ismét várakozó állapotba kerül.

A fenti szemléletmód hasznosságát a következők alkalmazásával mutatjuk be.

Ha egy telefonközpont hívásfeldolgozó rendszerében minden egyes híváshoz egy-egy SDL-processzort rendelünk, akkor a hívásfeldolgozás műveletsorát (SDL diagramját) elegendő egyetlen hívásra kidolgozni. Több hívás esetén csak az egyes SDL-processzorok megfelelő időben történő aktivizálásáról, és az egymás közötti üzenetváltások biztosításáról kell gondoskodnia egy felügyeleti rendszernek.

Fel kell ismerni azonban azt is, hogy egy tárolt programvezérlésű automatának az ilyen módon történő leírása csak mint egy élő szervezet csontváza fogható fel, amire feltapadnak azok az „izom-modulok" (programok), amik a tényleges működést biztosítják. Ennek a „csontváz"-nak a rendező ereje viszont vitathatatlan előny a moduláris programtervezésben.

4. Az SDL-processzor programozása

Ha az SDL-processzort realizálni akarjuk, szükségünk van egy olyan eljárásra, amivel a grafikusan megjelenő rendszertervet („működési menetrendet") egy valóságos processzor által értelmezhető formára tudjuk transzformálni.

Ezt a feladatot két lépésben oldjuk meg:

1. Megadunk egy szimbólum—szó megfeleltetést, amelyben az SDL kép szavakkal leírható.
2. Kidolgozunk egy fordító programot, ami az előzőek szerint előállított forrásszöveget a processzor gépi kódjaira transzformálja át.

A leíró nyelv az alábbi kulcsszavakat, és azok kombinációit használja.

STATE-INPUT konstrukció

Ez felel meg az SDL állapot és bemenet szimbólumaiból alkotott képnek.

A STATE kulcsszó mindig egy input-blokk sorozatot vezet be. A STATE utáni számmal azt kell megadni, hogy hány input-blokkot kívánunk szerepeltetni ebben az állapotban.

Egy input-blokk az INPUT kulcsszót tartalmazó sorral kezdődik, amit a bemenő jelzést azonosító megnevezés követ. Ezután :-tal elválasztva egy végrehajtó utasításnak kell állnia. (Ez lehet egy újabb STATE-INPUT vagy egy PROC-CASE konstrukció, egy ugró utasítás, vagy egy programcím.)

Az input-blokk további soraiban kell írni az INPUT kulcsszó nélkül azokat a végrehajtó utasításokat, amelyek ehhez az inputhoz tartoznak. Az utasítások leírásuk sorrendjében kerülnek végrehajtásra. Egy input-blokk tetszőleges számú utasítás sort tartalmazhat.

PROC-CASE konstrukció

Ez felel meg az SDL döntési sémájának.

Első sorában a PROC kulcsszó áll, amit annak a programnak a neve követ, ami ténylegesen végrehajtja a döntési műveletet. A név után álló szám mutatja, hogy a döntésnek hány lehetséges kimenetét kívánjuk definiálni. Minden egyes döntési alternatívához egy CASE-blokk tartozik. Egy CASE-blokk szerkezete azonos az input-blokkéval, csak itt az első sorban a CASE kulcsszónak kell állnia.

Vezérlés átadás (GO)

A GO ⟨címke vagy programnév⟩ utasítás sor hatására a vezérlés feltétel nélkül a GO kulcsszót követő címkére vagy programra adódik át. Címkék a forrásszöveg minden sorának első pozíciójától kezdve állhatnak.

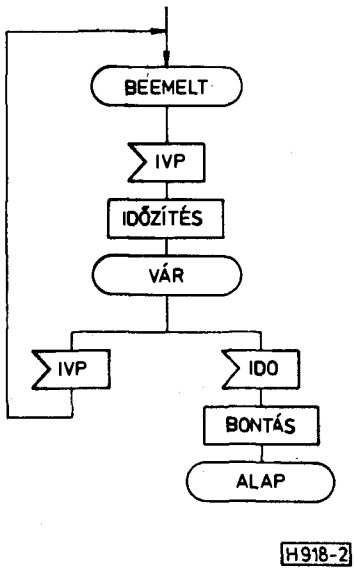
Visszatérés (RETURN)

A RETURN kulcsszó hatására az SDL-processzor úgy kerül várakozó állapotba, hogy kiindulóállapota nem változik meg (nem következik be állapotváltozás).

5. Direktívák

Fentieken túl a forrásszöveg tartalmaz olyan direktívákat, amelyek a fordító számára közölnek információt.

Ilyenek, az ORIGIN ⟨szám⟩, amivel azt adhatjuk meg, hogy milyen memóriacímtől kérjük a tárgyprogramot; az EQU, amivel értéket adhatunk a forrásszövegben szimbolikusan szereplő programneveknek; az END, ami a forrásszöveg végét jelzi.



2. ábra. A mintafeladat megoldásának SDL képe

6. Mintapélda

Az előbbiek szemléltetésére dolgoztuk ki az alábbi feladat megoldását.

Tervezni kell egy olyan SDL-processzort, ami azt figyeli, hogy a felemelt kézibeszélőjű állapotban levő készülék bontott-e?

A kézibeszélő visszahelyezése állapotváltozást hoz létre a központban a készülékhez tartozó ívponton. Egy ilyen esemény észlelésekor azonban nem mondhatjuk azt, hogy a készüléknél biztosan bontás történt, mert például így a kézibeszélő akaratlan visszaejtését is bontásnak értelmeznénk. Ezért azt a megoldást választottuk, hogy csak azt az esetet tekintjük bontásnak, amelyben az ilyen változást meghatározott ideig nem követi újabb állapotváltozás.

A feladat megoldásának SDL-képe a 2. ábrán látható.

Az SDL-processzor mindaddig a BEEMELT állapotban várakozik, amíg ívpontjának állapota meg nem változik. Az IVP input jel hatására időzítést kell indítani, és a processzor VAR állapotban várakozik a további jelzésekre. Ebben a VAR állapotban az automatát már két bemenő jelzés fogadására készítjük fel:

1. Újabb változás az ívponton (IVP), aminek hatására visszatér BEEMELT állapotba.
2. Az előzőekben indított időzítés letelik (IDO), ami megállapodásunk értelmében azt jelenti, hogy a készülék ténylegesen bontó állapotba került, így megindulhat a bontás folyamata.

Az SDL-kép alapján megírt forrásszöveget a 3. ábrán láthatjuk.

A protokoll szinte szóról szóra követi az SDL grafikus ábráját. Az EQU-val megadott változók azoknak a szubrutinoknak a memóriacímei, amelyek ténylegesen végrehajtják a működéshez szükséges akciókat. (Ezek másutt vannak leírva, itt csak behívásuk és aktivizálásuk történik meg.)

A fenti forrásnyelvre kialakított fordítónk olyan gépi kód sorozatot generál, amit az I 8085 mikroprocesszor közvetlenül értelmezni tud.

7. Hierarchikus SDL-proeesszorok

Egy rendszer SDL-képe megrajzolható úgy, hogy abban csak olyan belső állapotokat definiálunk, amelyeknek inputjai közvetlenül hardware-től kapott jelzések.

Ebben az esetben azonban az ábra áttekinthetetlenül naggyá válik, és sokszor ismétlődő, azonos részleteket tartalmaz. Előző példánkat idézve, minden olyan helyzetben, ahol a készülék kézibeszélőjének visszahelyezését kell figyelnünk, szerepeltetni kellene a mintapélda SDL-képét. Ez nem lenne gond akkor, ha minden visszahelyezést a bontás folyamatának kellene követnie. Ekkor egyszer leírnánk a visszahelyezés akciósorozatát, és erre a GO utasítással mindenünnen rá lehetne ugrani. A valóságos helyzet azonban más. Itt a visszahelyezést mindig azonos eljárással figyeljük, az azt követő akciónak azonban esetenként másnak kell lennie.

A probléma egyszerű megoldása az, hogy definiá-

```
0000 ********************************************************
0005 *       A LETEVEST FIGYELO SDL-PROCESSZOR              *
0010 ********************************************************
0015       ORIGIN 0A000H
0020 BEEMEL    STATE 1
0025                 INPUT IVP:IDOZITES
0030                           GO VAR
0035 VAR       STATE 2
0040                 INPUT IVP:GO BEEMEL
0045                 INPUT IDO:BONTAS
0050                           GO ALAP
0055 IDOZITES EQU 5678H
0060 BONTAS EQU 0ABCDH
0065 ALAP EQU 0
0070   END
```

3. ábra. Az SDL-kép alapján megírt forrásszöveg

lunk egy olyan SDL-processzort, ami csak a visszahelyezést figyeli. Ha ennek megtörténtét megállapította, nem konkrét műveletet hajt végre, hanem csak egy jelzést ad ki erről, esetenként meghatározott outputjára (úgynevezett software jelzés). Ezt egy másik SDL-processzor input jelzésként dolgozza fel, ekkor már a konkrét működtető program indítására.

Ezzel a módszerrel az SDL-processzoroknak egy hierarchiája alakítható ki. A rendszer legalacsonyabb szintjén állnak azok az SDL-processzorok, amelyek közvetlen hardware jelzéseket dolgoznak fel, és kimenetük software jelzés. A következő szinten vannak azok, amelyek már software jelzéseket is értelmezni tudnak, és közvetlen működtető programok indítására is alkalmasak. Úgy is lehet mondani, hogy a magasabb szinten álló SDL-processzorok konkrét szituációra tervezettek, míg az alacsonyabb szintűek általános alkalmazhatóságúak.

További példa lehet a számjegy-bevételezés megoldása különböző szolgáltatások esetén. Ekkor az alacsonyabb szinten elhelyezkedő SDL-processzor a beérkező vonali jelzések alapján bevételezi a számjegyeket, de további feldolgozásukról nem dönt, csak jelzést küld az őt kiértékelő SDL-processzornak, ami a szituációnak megfelelően van felkészítve. Amennyiben a hívószám egyszerű házi szám, akkor a felkapcsolás folyamatát indítja el, ha speciális szám, akkor annak lekezelésére indít műveleteket.

A jelzések érzékelésére és átadására rendszerünkben nincsenek meghatározott eljárások. Ezek tényleges megvalósítása azokkal a szubrutinokkal történik, amiket az SDL csontvázra építve esetenként megtervezünk.

8. Köszönetnyilvánítás

Végezetül a szerző köszönetet mond Dági Lajos és Szádeczky-Kardoss Tamás villamosmérnököknek, akik az ismertetett módszer kidolgozásában és alkalmazásában vitapartnerei, bírálói és segítői voltak.

IRODALOM

[1] CCITT ajánlások. Yellow Book, Vol. VI/7.

[2] *V. Giarrantina, M. Modesti:* An SDL into CHILL skeleton translation system. Fifth Int. Conf. on Software Engineering for Telecommunication Switching Systems. 1983. Lund, Southern Sweden.

[3] *Makay A.:* TPV telefonközpontok hívásfeldolgozó feladatainak programozása. Híradástechnika, XXXIV. évf. 1983. 1. sz.

[4] *Horváth I.:* Magyar fejlesztésű kis kapacitású digitális alközpontcsalád. Híradástechnika, XXXV. évf. 1984. 6. sz.