فصل ششم

رفع اشکال نرم‌افزاری

۶-۱- تشخیص اشکالات برنامه‌نویسی

یکی از مهمترین مواردی که همواره مد نظر برنامه نویسان بوده و هست تشخیص اشکالات نرم‌افزاری و نحوهٔ رفع اشکال در برنامه است. همانگونه که می‌دانید پس از نوشتن یک برنامهٔ کامپیوتری به هر زبان نیاز به کامپایل نمودن[1] برنامه و رفع اشکالات احتمالی آن است.

در پروگرامرهای PLC کامپایلری جهت این کار وجود ندارد، اما وجود اشکالات نرم‌افزاری را می‌توان در بخشی از حافظه به نام ISTACK[2] و BSTACK[3] جستجو نمود. این بخش، حافظهٔ داخلی CPU می‌باشد و محلی است که اطلاعاتی در مورد دستور نادرست یا به عبارت دیگر دستوری که باعث ایجاد وقفه در اجرای برنامه شده به دست می‌دهد. در این حالت PLC به حالت STOP[4] رفته، اطلاعات آخرین دستوری که باعث توقف برنامه شده در ISTACK باقی می‌ماند.

باید توجه داشت که تنها در حالتی می‌توان به ISTACK دست یافت که PLC در حالت RUN بوده، سپس به حالت STOP رفته باشد. برای عیب‌یابی نرم‌افزاری نیاز به یک سری اطلاعات است

---

1 - COMPILE 2 - INTERRUPT STACK 3 - BLOCK STACK

4 - STOP MODE

که توسط PLC در اختیار ما قرار می‌گیرد.

به محض اینکه یک بلوک توسط برنامه‌نویس ایجاد می‌شود (... ، FB ، PB ، OB) در حافظهٔ PLC، ۵ کلمه (WORD) توسط زبان STEP 5 اشغال می‌شود.

- در کلمهٔ اول عدد 7070_H قرار می‌گیرد، که به آن، کلمهٔ همزمان کننده یا Synchronization Word می‌گویند. بنابراین با دستیابی به Memory Map یا نقشهٔ حافظه و مشاهدهٔ عدد 7070_H در می‌یابیم که این نقطه، محل ایجاد و باز شدن یک بلوک است.

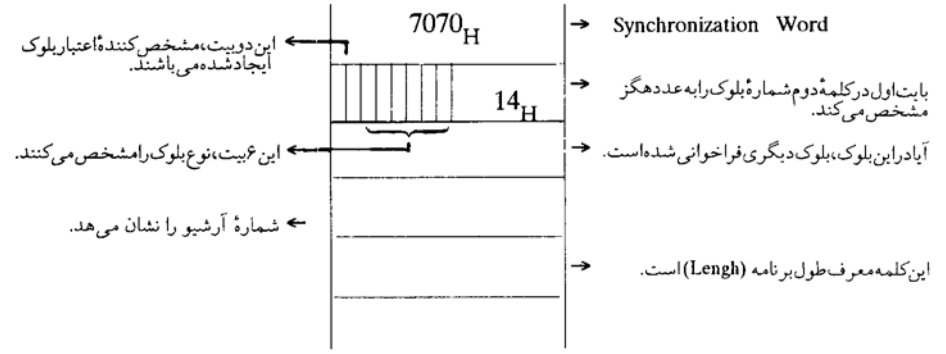
- در کلمهٔ دوم شمارهٔ بلوک، نوع و اعتبار بلوک قرار می‌گیرد. در این کلمه، دو بایت وجود دارد که بایت اول معرف شمارهٔ بلوک به عدد هگز می‌باشد و بایت بعدی را به بیت‌های جداگانه‌ای اختصاص می‌دهد. در دو بیت اول این بایت اعتبار بلوک و در ۶ بیت دیگر نوع بلوک مشخص می‌شود. به عنوان مثال اگر در ۶ بیت مذکور عدد $۴_{(۱۰)}$ را به هگز قرار دهد بدین معنی است که بلوک ایجاد شده از نوع PB است.

- کلمهٔ سوم مشخص کنندهٔ آن است که آیا در بلوک ایجاد شده، بلوک دیگری که معمولاً از نوع FB است فراخوانی شده است یا خیر؟

- کلمهٔ چهارم معرف آن است که آیا این بلوک دارای شمارهٔ آرشیو است یا خیر؟ زیرا برای هر بلوک می‌توان شمارهٔ آرشیو مشخص نمود.

- کلمهٔ پنجم طول برنامه را برحسب بایت مشخص می‌کند.

به این ۵ کلمه که توسط S5 در حافظه اشغال می‌شوند Block Header می‌گویند. از کلمهٔ ششم به بعد دستورات موجود در بلوک ایجاد شده که به کد ماشین تبدیل شده‌اند دیده می‌شوند. در شکل ۶-۱ نمونه‌ای از Block Header را می‌بینید:



شکل ۶-۱ : نقشه حافظه و Block Header ایجاد شده در آن

به دلیل اینکه دستورالعمل نادرست در ISTACK و BSTACK به صورت کد ماشین به برنامه‌نویس ارائه می‌شود ابتدا به بررسی چگونگی تبدیل کد ماشین به دستورات S5 می‌پردازیم.

۶-۲- تبدیل کد ماشین به دستورات STEP 5

در جدول ضمیمهٔ ۳، کدهای ماشین استفاده شده در PLC زیمنس آورده شده است. این کدها را می‌توان به چند دستهٔ عمومی تقسیم نمود.

- دسته‌ای از کدها که احتیاج به تغییر و تبدیل ندارند مانند کدهای 6500 و 6501 و ... که به ترتیب معرف دستورات BE و BEU می‌باشند. در این دستورات هیچ عملوندی یافت نمی‌شود.

- دسته‌ای از کدها که نیاز به تغییر و تبدیل به عملکرد و عملوند و همچنین آدرس عملوند دارند. روش برخورد با این کدها به صورت زیر است:

با جستجو در جدول حاوی کدهای ماشین، بزرگترین کد موجود در جدول را که به هنگام تفاضل از کد مورد نظر نیاز به بیت قرضی[1] نداشته باشد یافته، حاصل تفریق را به دست می‌آوریم. کُد متناظر موجود در جدول شامل قسمتی از دستورالعمل نادرست است و به عنوان مثال به صورت A I ... می‌باشد ولی شماره و آدرس عملوند در آن یافت نمی‌شود. حاصل تفریق، مشخص کنندهٔ آدرس عملوند است. در مثالهای زیر سعی شده تا تمام موارد ممکن بررسی شود تا خواننده در عیب‌یابی نرم‌افزاری و تبدیل کدهای ماشین به دستورات S5 تسلط کافی پیدا کند.

مثال ۶-۱ : کد ماشین 6500 را به دستورالعمل S5 تبدیل کنید.

با مقایسهٔ کد داده شده و کدهای موجود در جدول ملاحظه می‌کنید که این کد مربوط به دستورالعمل BE است. از آنجایی که این دستورالعمل فاقد عملوند می‌باشد در جدول بدون هیچگونه عملوند دیده می‌شود. در صورتی که این کد، کد دستورالعمل نادرست باشد بدین معنی است که احتمالاً در حین برنامه‌نویسی این دستور از قلم افتاده است.

مثال ۶-۲ : کُد ماشین C000 را به دستورالعمل S5 تبدیل کنید.

همان‌طور که گفته شد با جستجو در جدول ضمیمهٔ ۳ بزرگترین کُد ماشین را که به هنگام تفریق

---

1 - Borrow Bit

از کد C000 نیاز به بیت قرضی نداشته باشد می‌یابیم. سپس با انجام عملیات تفریق و به دست آوردن حاصل تفریق، شمارهٔ عملوند و آدرس آن را به دست می‌آوریم. بزرگترین کد ماشین موجود در جدول که به هنگام تفاضل نیاز به بیت قرضی نداشته باشد همان کُد C000 است که معرف دستور A I ... می‌باشد. آدرس عملوند از حاصل تفریق به دست می‌آید.

C000 -
C000 ⟶ A I 0.0 ← مقایسه در جدول و یافتن کد مذکور
00.00
شماره بایت ← ← شماره‌بیت

بنابراین دستورالعمل نادرست A I 0.0 بوده است.

مثال ۶-۳: دستورالعمل متناظر با کد ماشین C010 را پیدا کنید.

بزرگترین کد موجود در جدول که به هنگام تفاضل از کد C010 نیاز به بیت قرضی ندارد C000 است. آدرس عملوند نیز از حاصل تفریق به دست می‌آید.

C010 -
C000 ⟶ A I 16.0 ← مقایسه در جدول و یافتن کد مذکور
00.10
شماره‌بیت ← ← شمارهٔ بایت به هگز: 1 × 16 + 0 = 16

مثال ۶-۴: دستورالعمل متناظر با کد ماشین CA85 کدام است؟

با مقایسه در جدول ملاحظه می‌کنید که کد C880 بزرگترین کد ماشین است که به هنگام تفاضل از کد CA85 نیاز به بیت قرضی ندارد. این کد معرف ... O Q می‌باشد.

CA85 -
C880 ⟶ O Q 5.2 ← مقایسه در جدول و یافتن کد مذکور
02.05
شماره‌بیت به هگز: 0 × 16 + 2 = 2 ← ← شماره بایت به هگز: 0 × 16 + 5 = 5

مثال ۶-۵: کد ماشین 343B را به دستورالعمل متناظر آن در S5 تبدیل کنید.

بزرگترین کد موجود در جدول کد 3400 بوده که متناظر با دستور ... SP T می‌باشد. شمارهٔ تایمر از حاصل تفریق به دست می‌آید.

343B -
3400 ⟵ مقایسه در جدول و یافتن کد مذکور ⟵ SP T 59
003B
3 × 16 + 11 = 59

مثال ۶-۶: دستورالعمل متناظر با کد ماشین 7582 را بیابید.

با مقایسه در جدول به کد 7500 دست خواهیم یافت که معرف پرش غیرشرطی به بلوک برنامه به صورت ... JU PB می‌باشد. شمارهٔ PB مورد نظر نیز از حاصل تفریق به دست می‌آید.

7582 -
7500 ⟵ مقایسه در جدول و یافتن کد مذکور ⟵ JU PB 178
0082
8 × 16 + 2 = 178

مثال ۶-۷: کد ماشین F823 را به دستورالعمل متناظر آن در S5 تبدیل کنید.

کد F800 بزرگترین کد ماشین است که به هنگام تفاضل از کد F823 نیاز به بیت قرضی ندارد. این کُد معرف دستور ... A T می‌باشد و شمارهٔ تایمر نیز از حاصل تفریق به دست می‌آید.

F823 -
F800 ⟵ مقایسه در جدول و یافتن کد مذکور ⟵ A T 35
0023
2 × 16 + 3 = 35

مثال ۶-۸: دستورالعمل متناظر با کد ماشین 861E را بیابید.

با مقایسه در جدول به کد 8000 دست خواهیم یافت. این که معرف دستورالعمل ... A F می‌باشد که آدرس فلگ از حاصل تفریق به دست می‌آید.

861E -
8000 ⟵ مقایسه در جدول و یافتن کد مذکور ⟵ A F 30.6
06.1E
30 = 14 + 16 × 1 : شمارهٔ بایت به هگز ⟵ ⟵ شمارهٔ بیت به هگز : 6 = 6 + 16 × 0

حال که با روش تبدیل کد ماشین به دستورالعمل متناظر با آن در S5 آشنا شدیم به بررسی ISTACK و BLOCK STACK می‌پردازیم. در پروگرامرهای مختلف، صفحهٔ نمایش ISTACK نیز متفاوت می‌باشد ولی اصول کار همگی آنها تقریباً یکسان است. در جداول ۶-۱ و ۶-۲ صفحهٔ

نمایش ISTACK در پروگرامر PG 605 و PG 615U نشان داده شده است. قسمتهایی که با خطوط پر رنگ‌تر مشخص شده‌اند نشان دهندهٔ علت ایجاد اشکال و همچنین آدرس دستورالعمل نادرست می‌باشد. در جداول ضمیمهٔ ۴ این اشکالات به صورت مشروح آورده شده است.

جدول ۶-۱ : صفحهٔ نمایش ISTACK در پروگرامرهای PG 615 U , PG 605

Bit / Byte	7	6	5	4	3	2	1	0	Absolute addr.	System data word (RS)
1			BST SCH	SCH TAE	ADR BAU	SPA BBR			EA0A	SD 5
2	CA-DA	CE-DA		REM AN					EA0B	
3	STO ZUS	STO ANZ	NEU STA		BAT PUF		BARB	BARB END	EA0C	SD 6
4		UA FEHL				AF			EA0D	
5	ASP NEP	ASP NRA	KOPF NI		ASP NEEP				EA0E	SD 7
6	KEIN AS	SYN FEH	NINEU					UR LAD	EA0F	
7	IRRELEVANT									
8	IRRELEVANT									
9	STOPS		SUF	TRAF	NNN	STS	STUEB	FEST	EBAC	SD 214 (UAW)
10	NAU	QVZ	KOLIF	ZYK	SYSFE	PEU	BAU	ASPFA	EBAD	
11									EBAA	SD 213
12	ANZ1	ANZ0	OVFL		OR	STA TUS	VKE	ERAB	EBA8	
13	6th nesting level					OR	VKE	FKT	EBA8	SD 212
14	IRRELEVANT								EBA9	
15	4th nesting level					OR	VKE	FKT	EBA6	SD 211
16	5th nesting level					OR	VKE	FKT	EBA7	

جدول ۶-۲ : صفحهٔ نمایش ISTACK در پروگرامرهای PG 605 , PG 615 U

(ادامه جدول ۶-۱)

Byte \ Bit	7	6	5	4	3	2	1	0	Abso-lute addr.	System data word (SD)
17	2nd nesting level					OR	VKE	FKT	EBA4	SD 210
18	3rd nesting level					OR	VKE	FKT	EBA5	
19	Nesting depth (0 to 6)								EBA2	SD 209
20	1st nesting level					OR	VKE	FKT	EBA3	
21	Start address of the data block (high)								EBA0	SD 208
22	Start address of the data block (low)								EBA1	
23	Block stack pointer (high)								EB9E	SD 207
24	Block stack pointer (low)								EB9F	
25	Step address counter (high)								EB9C	SD 206
26	Step address counter (low)								EB9D	
27	Statement register (high)								EB9A	SD 205
28	Statement register (low)								EB9B	
29	ACCUM 2 (high)								EB98	SD 204
30	ACCUM 2 (low)								EB99	
31	ACCUM 1 (high)								EB96	SD 203
32	ACCUM 1 (low)								EB97	

در جداول زیر صفحات نمایش ISTACK و بیت‌های کنترلی را بر روی پروگرامر مشاهده می‌کنید. در صفحهٔ نمایش CONTROL BITS قسمتهایی که با حروف پر رنگ‌تر مشخص شده‌اند نشان دهندهٔ یکی از احتمالات وجود خطا می‌باشد.

جدول ۶-۳ : صفحهٔ نمایش بیت‌های کنترلی در پروگرامرهای مذکور

CONTROL BITS								Absolute address	System data word
NB	PBSSCH	BSTSCH	SCHTAE	ADRBAU	SPABBR	NAUAS	QUITT	EA0A	SD5
CA-DA	CE-DE	NB	REMAN	NB	NB	NB	NB	EA0B	
STOZUS	STOANZ	NEUSTA	NB	BATPUF	NB	BARB	BARBEND	EA0C	SD6
NB	UAFEHL	MAFEHL	EOVH	NB	AF	NB	NB	EA0D	
ASPNEP	ASPNRA	KOPFNI	PROEND	ASPNEEP	PADRFE	ASPLUE	RAMADFE	EA0E	SD7
KEINAS	SYNFEH	NINEU	NB	NB	NB	SUMF	URLAD	EA0F	

جدول ۶-۴ : صفحهٔ نمایش ISTACK در پروگرامرهای مذکور

INTERRUPT STACK														Abso-lute addr.	System data word
DEPTH: 01															
BEF-REG: 0000 BST-STP: EB07		SAZ: E30A Baust.-NR.: 1 REL-SAZ: 0000			DB-ADR: 0000 DB-NR.:									EB9A -EBA0	SD205-208
AKKU1: FFF1		AKKU2: 00FF												EB96-EB98	SD203-204
BRACKETS:		KE1	000	KE2	000	KE3	000	KE4	000	KE5	000	KE6	000	EBA2-EBA8	SD209-212
RESULT DISPLAY:		ANZ1	ANZ0	OVFL	CARRY			ODER	STATUS	VKE	ERAB			EBAA	SD213
CAUSE OF FAULT:		STOPS			SUF	TRAF	NNN		STS	STUEB				EBAC	SD214 (UAW)
		NAU	QVZ			ZYK			PEU	BAU	ASPFA			EBAD	

در جداول ۶-۵ و ۶-۶ نیز صفحات نمایش بیت‌های کنترلی و ISTACK در پروگرامر 685U آورده شده است. به شباهتهای موجود بین جداول مذکور و جداول ۶-۳ و ۶-۴ توجه کنید.

جدول ۶-۵ : صفحهٔ نمایش بیت‌های کنترلی در پروگرامر 685U

C O N T R O L BITS

NB	PBSSCH	BSTSCH	SCHTAE	ABRBAU	SPABBR	NAUAS	QUITT
NB	NB	NB	REMAN	NB	NB	NB	NB
STOZUS X	STOANZ X	NEUSTA	NB	BATPUF X	NB	BARB	BARBEND
NB	UAFEHL	MAFEHL	EOVH	NB	AF	NB	NB
ASPNEP	ASPNRA	KOPFNI	PROEND	ASPNEEP	PADRFE	ASPLUE	RAMADFE
KEINAS	SYNFEH	NINEU	NB	NB	NB	SUMF	URLAD

جدول ۶-۶ : صفحهٔ نمایش ISTACK در پروگرامر 685U

I N T E R R U P T S T A C K

D E P T H: 01

OP-REG :	3200	SAC:	9066	DB-ADD:	0000
BLK-STP:	EB07	PB-NO :	25	DB-NO.:	
		REL-SAC:	0004		

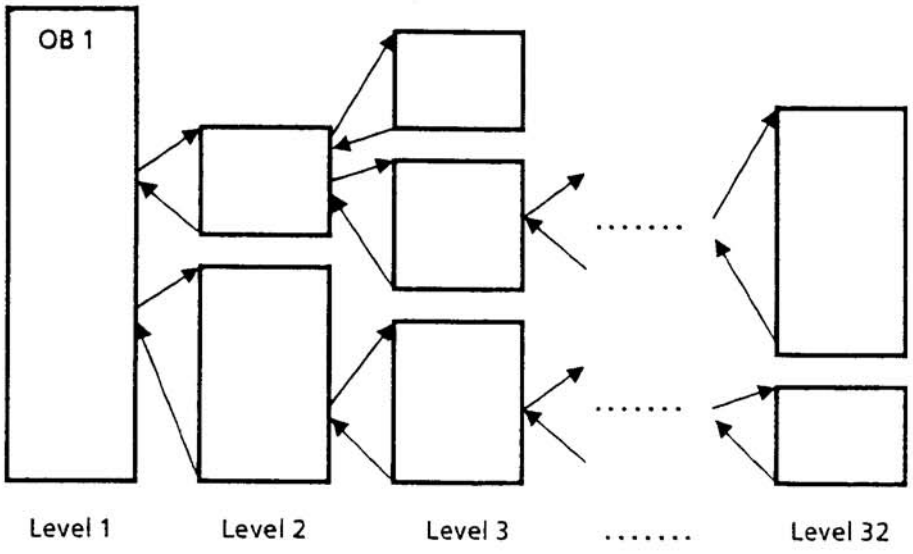
ACCU1: 1388 ACCU2: 05D8

CONDITION CODE :	CC1	CC0	OVFL	CARRY	OR	ERAB X
	STATUS	RLO X				
CAUSE OF INTERR.:	STOPS	NB	SUF	TRAF X	NNN	STS
	STUEB	NAU	QVZ	ZYK	PEU	BAU
	ASPFA					

اکنون به توضیح برخی از بخشهای مهم در جدول ۶-۴ می‌پردازیم. لازم به ذکر است که توضیحات ارائه شده برای موارد مشابه در جدول ۶-۶ نیز صادق است.

DEPTH : شماره‌ای که مقابل این بخش از صفحهٔ ISTACK قرار می‌گیرد معرف شمارهٔ سطح (LEVEL) بلوک‌ها در یک برنامه است. در جدول ۶-۴ منظور از DEPTH 1 اولین بلوک در اجرای برنامه یعنی OB 1 می‌باشد، بنابراین دستورالعمل نادرست در OB 1 وجود دارد. در شکل ۶-۲ سطوح (LEVEL) مختلف در برنامه‌نویسی نشان داده شده است.

لازم به ذکر است که تعداد سطوح یا تعداد پرشهای افقی در یک برنامه در PLCهای گوناگون، متفاوت است. در اینجا فرض بر آن است که آخرین سطح پرش 32 LEVEL می‌باشد.



شکل ۶-۲ : تعداد سطوح یا پرشهای افقی مجاز در یک برنامه

BEF-REG : کُد دستورالعمل نادرست را به صورت کد ماشین نشان می‌دهد. این کُد ماشین متناظر با دستورالعمل نادرستی است که باعث ایجاد توقف در اجرای برنامه شده است.

SAZ: آدرس محلی از حافظه است که دستورالعمل نادرست در آن قسمت از حافظه قرار گرفته است.

DB-NR , DB-ADR : در صورتی که در بلوکی که حاوی دستورالعمل نادرست است اطلاعاتی از بلوک DB فراخوانده شده باشد DB و آدرس DB مذکور در حافظه را نشان می‌دهد.

AKKU 1 , AKKU 2:محتویات دو انبارک را در لحظهٔ ایجاد وقفه (STOP) نشان می‌دهد.

۶-۳- دستیابی به دستورالعمل نادرست با استفاده از ISTACK

همانگونه که عنوان شد در جدول ISTACK بخش BEF-REG حاوی کُد دستورالعمل نادرست می‌باشد که به صورت کُد ماشین بر روی صفحهٔ نمایش ظاهر می‌گردد. اکنون برای درک بهتر چگونگی دستیابی به دستورالعمل نادرست، جدول ۶-۴ را در نظر می‌گیریم. در بخش

DEPTH شمارهٔ 01 درج شده است و این شماره نشان دهندهٔ شمارهٔ سطح بلوکی است که در آن بلوک دستورالعمل نادرست وجود دارد. با توجه به شمارهٔ 01 و قرار داشتن بلوک OB 1 در این سطح می‌توان گفت که دستورالعمل نادرست در OB 1 قرار دارد.

در بخش BEF-REG کد 0000 دیده می‌شود. با مراجعه به جداول موجود در ضمیمهٔ ۳ ملاحظه می‌کنیم که کد مذکور مربوط به دستورالعمل NOP 0 است. بنابراین دستور نادرست NOP 0 در بلوک OB 1 باعث ایجاد وقفه شده است.

با دقت در جدول ۶-۴ ملاحظه می‌کنید که در بخش DB-ADR و DB-NR عدد 0000 وجود دارد که نشان دهندهٔ عدم فراخوانی بلوک DB در برنامه می‌باشد.

در جدول ۶-۶ نیز در بخش DEPTH عدد 01 دیده می‌شود. بنابراین دستورالعمل نادرست در OB 1 قرار دارد. در بخش OP-REG که متناظر با بخش BEF-REG در جدول ۶-۴ می‌باشد کد 3200 وجود دارد که با مراجعه به جدول ضمیمه ملاحظه می‌کنید که مربوط به دستورالعمل ... L DW می‌باشد. با توجه به توضیحات ارائه شده، شمارهٔ DW به روش زیر به دست می‌آید.

3200 - کد موجود در OP-REG←
3200 کد موجود در جدول←

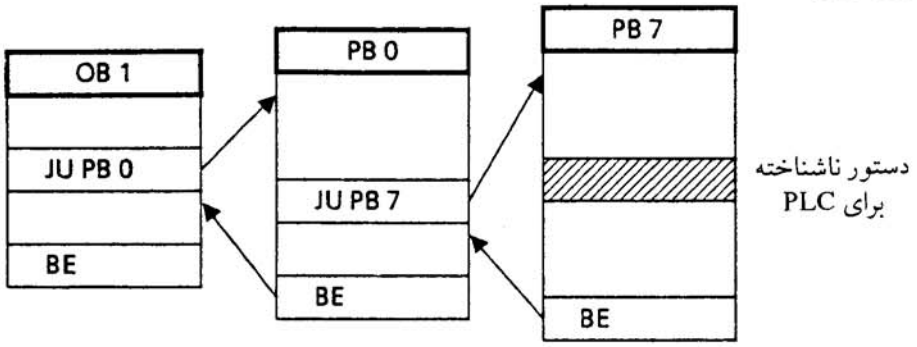
0000 ⟶ شمارهٔ DW

بنابراین دستورالعمل نادرست L DW 0 ، می‌باشد که در OB 1 قرار دارد. با دقت در جدول ۶-۶ ملاحظه می‌کنید که در بخش DB-ADD و DB-NO عدد 0000 وجود دارد که نشان دهندهٔ عدم فراخوانی بلوک DB در این برنامه است. بنابراین واضح است که وقفه بوجود آمده به دلیل بارگذاری یک کلمه (DW) از بلوکی است که در این برنامه فراخوانی نشده است.

۶-۴- روش دیگربرای دستیابی به دستورنادرست با استفاده از ISTACK

در توضیحات جداول ۶-۴ و ۶-۶ عنوان شد که بخش SAZ یا SAC در صفحهٔ نمایش ISTACK حاوی آدرس محلی از حافظه است که PLC قبل از رفتن به حالت STOP در آن محل قرار داشته است. بنابراین به کمک این آدرس می‌توان دستور نادرست را یافت. مثال زیر این مطلب را روشن می‌سازد.

مثال ۶-۹ : در شکل زیر یک برنامه راکه شامل ۳ بلوک PB 7 و PB 0 و OB 1 می‌باشد مشاهده می‌کنید. در PB 7 یک دستور ناشناخته برای PLC وجود دارد که باعث ایجاد وقفه در اجرای برنامه شده است.

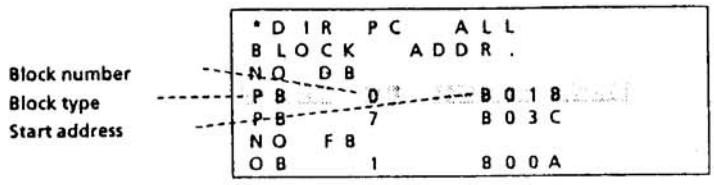


شکل ۶-۳ : این برنامه شامل سه بلوک است که یکی از بلوک‌ها حاوی دستوری ناشناخته برای PLC می‌باشد

در حین اجرای برنامه به محض اینکه PLC به دستور نادرست موجود در PB 7 برسد اجرای برنامه متوقف شده، PLC به حالت STOP می‌رود. در این حالت پیغام NNN ظاهر می‌گردد. این پیغام خطا معرف آن است که این سطر قابل پردازش و تعریف توسط CPU نمی‌باشد.

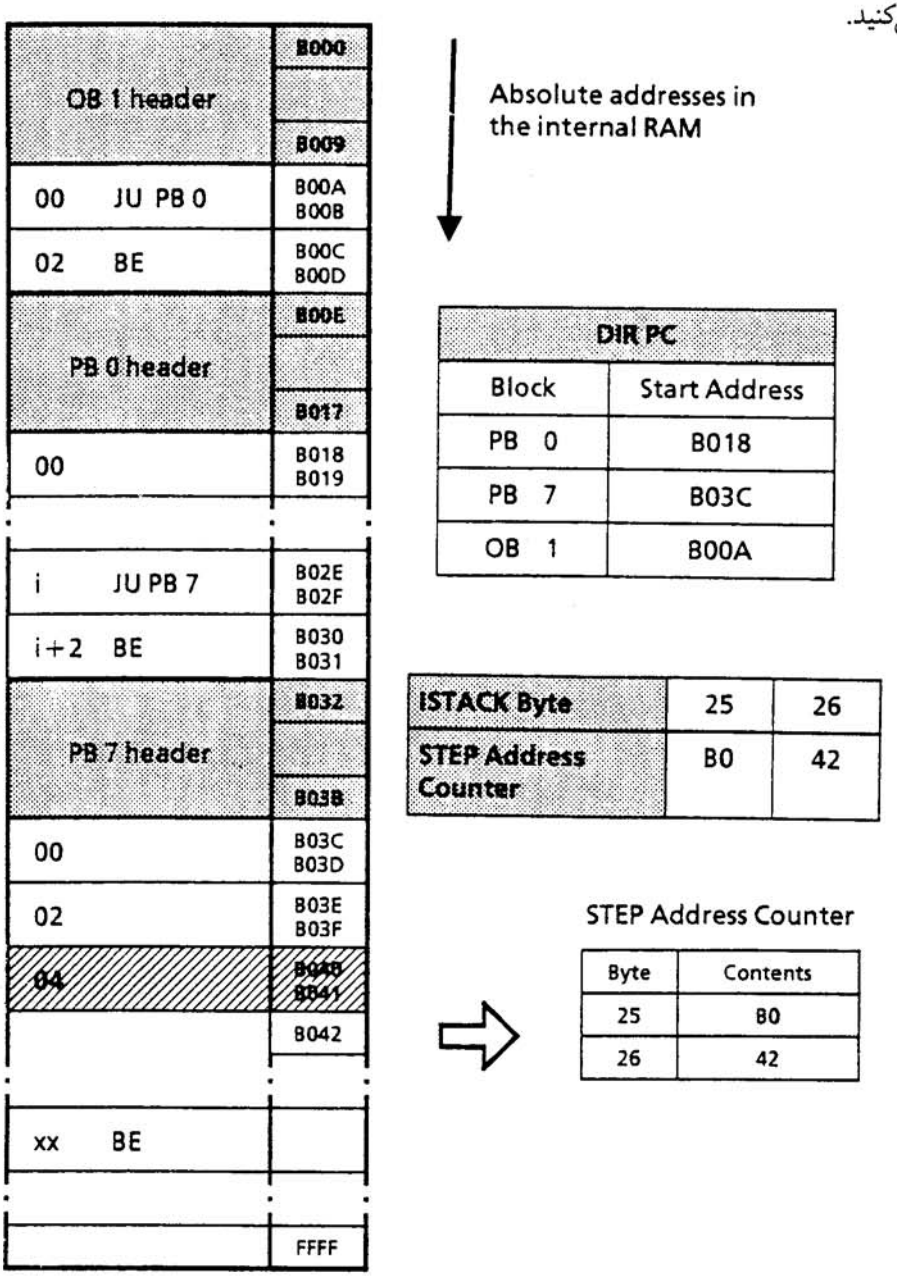
در بخش SAZ آدرس دستور نادرست در حافظه داده می‌شود. در زبان برنامه‌نویسی S5 دستور دیگری وجود دارد که به صورت PC DIR بوده، با اجرای آن می‌توان آدرس شروع هر بلوک در برنامه را بر روی صفحهٔ PG مشاهده نمود. بنابراین می‌توان با به دست آوردن تفاضل آدرس SAZ و آدرس شروع بلوکی که در آن، دستور نادرست وجود دارد به سطری که حاوی دستورالعمل نادرست می‌باشد دست یافت. البته این روش تنها در مورد پروگرامر 605U PG معتبر است.

در شکل ۶-۴ صفحهٔ نمایش دستور PC DIR بر روی پروگرامر 615U PG دیده می‌شود.



شکل ۶-۴ : صفحهٔ نمایش PC DIR در پروگرامر PG 615U

در شکل ۶-۵ نحوهٔ دستیابی به سطری از برنامه که حاوی دستور نادرست می‌باشد را ملاحظه می‌کنید.



شکل ۶-۵ : نحوهٔ دستیابی به شمارهٔ سطر محتوی دستورالعمل نادرست

با مقایسهٔ دو آدرس موجود در بخش SAZ یعنی B042 و آدرس شروع بلوک PB 7 یعنی B03C در می‌یابیم که دستورالعمل نادرست در این بلوک قرار دارد.

محاسبهٔ شمارهٔ سطر محتوی دستورالعمل نادرست: B042 − B03C

0006 ⟵ سطرششم ازبرنامهٔ PB 7 باعث توقف اجرای برنامه شده‌است.

۶-۵- دستیابی به دستورالعمل نادرست با استفاده از BSTACK

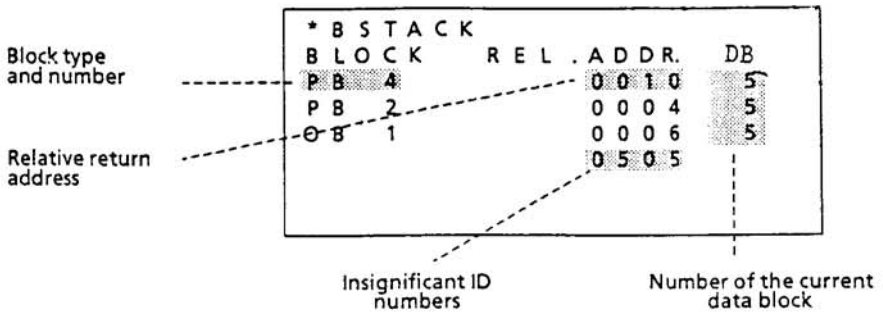
همان‌گونه که در فصول گذشته عنوان شد به هنگام پرش از یک بلوک به بلوک دیگر سه دسته اطلاعات زیر در BSTACK ذخیره می‌شوند:

- شمارهٔ بلوک DB که قبل از JUMP یا STOP معتبر بوده است.
- آدرس سطری از برنامه که هنگام بازگشت بایستی اجرای برنامه از آن سطر دنبال گردد.
- شماره و نوع بلوکی که از آن JUMP یا STOP انجام گرفته است.

با استفاده از BSTACK می‌توان به اطلاعات مذکور دست یافت. BSTACK حاوی وضعیت BLOCK STACK در زمانی است که در اجرای برنامه وقفه‌ای ایجاد شده است.

مثال ۶-۱۰: در طول اجرای یک برنامه، وجود دستوری در FB 2 باعث ایجاد وقفه و رفتن PLC به حالت STOP شده است. در این حالت بر روی صفحهٔ PG پیغام خطای TRAF ظاهر می‌شود که مربوط به فراخوانی نادرست اطلاعات از DB 5 می‌باشد.

در شکل زیر صفحهٔ نمایش BSTACK را در پروگرامر PG 615 می‌بینید.

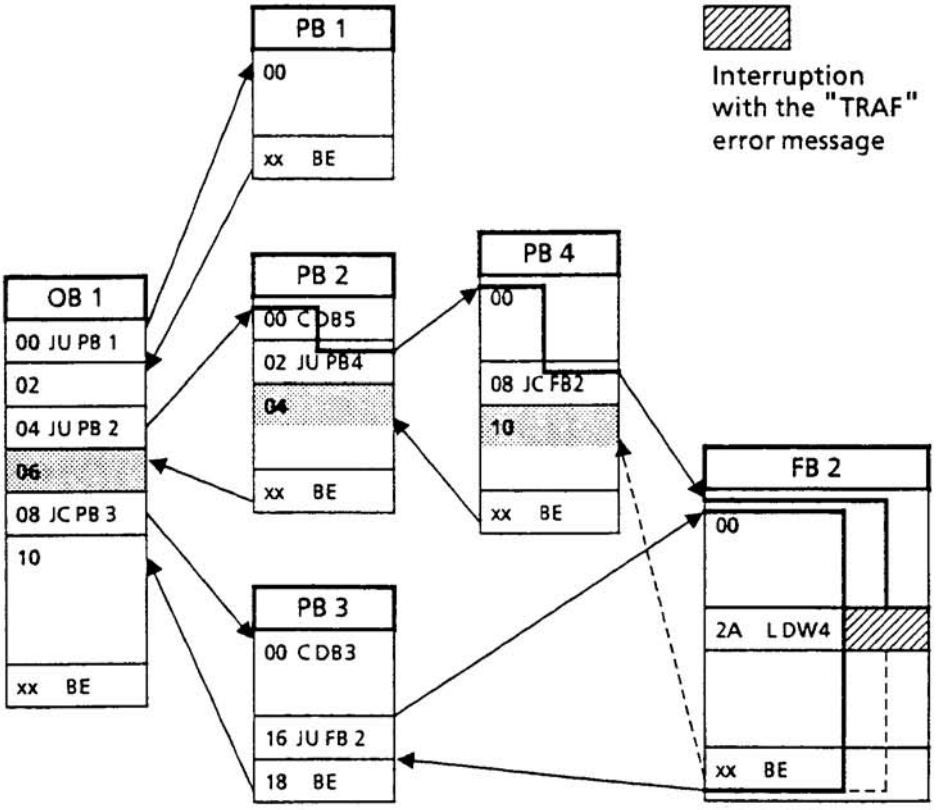


شکل ۶-۶: صفحهٔ نمایش BSTACK در پروگرامر PG 615

در شکل ۶-۶ ملاحظه نمودید که در طول اجرای برنامه به صورت دنبال OB 1→ PB 2→ PB 4، اطلاعات بلوک DB 5 به صورت نادرست خوانده شده است و یا اینکه در DB 5 اشکال ساختاری وجود دارد.

البته روش ذکر شده در مورد تمام پروگرامرها به استثنای PG 605 معتبر است.

در شکل ۶-۷ چگونگی روند اجرای برنامه تا زمانی که PLC به حالت STOP می‌رود نشان داده شده است.



شکل ۶-۷ : روند اجرای برنامهٔ مثال ۶-۱۰ تا زمان ایجاد وقفه در اجرای برنامه