ČESKÁ A SLOVENSKÁ FEDERATIVNÍ REPUBLIKA
(19)

FEDERÁLNÍ ÚŘAD PRO VYNÁLEZY

ZVEŘEJNĚNÁ PŘIHLÁŠKA VYNÁLEZU

(12)

(21) 00936-91.G

(13) A3

5(51) G 06 F 9/30, 9/00, 12/00

(22) 04.04.91

(32) 04.05.90

(31) 90/519384

(33) US

(40) 17.12.91

(71) International Business Machiness Corporation, Armonk, New York, US

(72) Blaner Bartholomew, Newark Valley, New York, US
Vassiliadis Stamatis, Vestal, New York, US

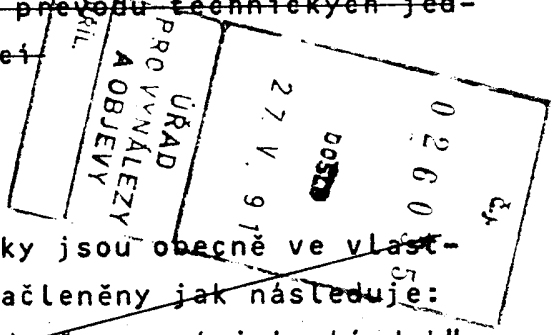
(54) Způsob zpracování sledu instrukcí a zařízení k provádění tohoto způsobu

(57) Při zpracování jsou vyhledávány třídy instrukcí, které je možno provést paralelně bez blokování závislými daty nebo závislými technickými prostředky. Bez ohledu na jejich původní sekvenci jsou jednotlivé instrukce kombinovány s jednou nebo větším počtem jiných jednotlivých instrukcí pro vytvoření složené instrukce, která eliminuje vzájemná blokování. Řídicí informace je připojena k identifikaci informace vztahující se na provedení složených instrukcí. Výsledkem je tok skalárních instrukcí složených nebo seskupených dohromady před dobou dekódování instrukce, takže jsou již označeny a identifikovány pro selektivní současné paralelní provedení prováděcími jednotkami. Složením se nemění výsledky cílového kódu a stávajícími programy je uskutečněno zlepšení výkonnosti, zatímco je dodržena slučitelnost s předchozími realizovanými systémy, pro které byly vytvořeny původní sledy instrukcí.

PV 936-91G

Způsob zpracování sledu instrukcí a zařízení k provádění tohoto způsobu

~~Strojová architektura způsobilá k převodu technických jednotek v souboru složených instrukcí~~



~~Analogické přihlášky~~

~~Následující příbuzné přihlášky jsou obecně ve vlastnictví právního nástupce a jsou začleněny jak následuje: "Technické vybavení počítačů se sdruženou závislostí dat" podáno 4. 4. 1990, ser. No. 07/504,910; a "Universální technika složení instrukčních úrovní paralelních procesorů" /EN990019/ podáno 4. 5. 1990, ser. No. 07/519,382.~~

Oblast techniky

Vynález se týká obecně paralelního zpracování instrukcí počítačem a zejména se týká zpracování toku instrukcí pro identifikaci těch instrukcí, které lze vydávat a vykonávat paralelně ve specifickém uspořádání systému počítače.

Dosavadní stav techniky

Pojetí paralelního provádění instrukcí pomohlo zvětšit výkon počítačových systémů. Paralelní provádění je založeno na existenci samostatných funkčních jednotek, které mohou vykonávat současně dvě nebo více stejných nebo různých instrukcí.

Jinou používanou technikou pro zvýšení výkonu systémů počítače je zřetězené zpracování. Zřetězené zpracování je obecně docíleno rozdělením funkce vykonávané počítačem na nezávislé dílčí funkce a zařazením samostatné části technického vybavení nebo stupně k vykonávání každé dílčí funkce. Každý stupeň je definován pro včasné zabrání jednoho taktu. Zřetězené zpracování poskytuje formu paralelního zpracování, neboť je možno vykonávat vícenásobné instrukce současně. Ideálně lze vsunout jednu novou

instrukci do zřetězeného vedení v jednom cyklu, přičemž každá instrukce v zřetězeném vedení je v různém stupni provádění. Operace je analogická překládání z jazyka symbolických adres, s řadou instancí zpracování součinu v různých stupních kompletace.

Mnohdy však úspěšnost paralelního provedení a/nebo zřetězeného zpracování není docíleno vlivem zpoždění způsobeným vzájemným blokováním během přenosu dat a vlivem vzájemného blokování v závislosti na přístrojové technice. Příklad vzájemného blokování v závislosti na datech je tak zvané blokování "zápis-čtení", kde první instrukce musí napsat svůj výsledek dříve než druhá instrukce může být čtena a následně použita. Příklad vzájemného blokování technickými prostředky je tam, kde musí první instrukce použít zvláštní složku technického vybavení a druhá instrukce musí rovněž použít tutéž zvláštní složku technického vybavení.

Jedním z dříve používaných technických postupů k vyloučení vzájemného blokování /někdy zvané zřetězené hazardy/ je dynamické rozvrhování. Dynamické rozvrhování je založeno na skutečnosti, že vsunutím speciálního technického vybavení je možno přeřadit instrukční posloupnosti potom co byly vydány do zřetězeného provedení.

Bylo učiněno také několik pokusů o zvětšení účinnosti pomocí tak zvaného statického rozvrhování, které je provedeno dříve než instrukční tok je vyvolán z paměti k výpočtu. Statické rozvrhování je docíleno pohybem kódu a tím přeřazením instrukčního sledu před výpočtem. Toto přeřazení vytvoří ekvivalentní tok instrukcí, který mnohem více využije technického vybavení pomocí paralelního zpracování. Takové statické rozvrhování je typicky provedeno v době kompilace. Přeřazené instrukce však zůstanou ve svém původním tvaru a běžné paralelní zpracování stále vyžaduje nějaký tvar dynamického určení právě před pro-

vedením instrukcí za účelem rozhodnutí zda vykonat dvě nejbližší instrukce sériově nebo paralelně.

Taková technika rozvrhování může zlepšit celkovou výkonnost zřetězeného počítače, ale nemůže sama uspokojit trvalé stávající požadavky na zvětšení výkonnosti. V tomto ohledu mnohé moderní návrhy na universální výpočet se týkají využití instrukční úrovně mimo to, které bylo dosaženo zřetězením. Například další paralelnost instrukční úrovně byla docílena explicitně vydáním vícenásobných instrukcí v jednom cyklu tak zvanými superskalárními stroji, spíše než implicitně dynamickým rozvrhováním jednotlivých instrukcí nebo vektorovými stroji. Název superskalární pro stroje, které vydávají vícenásobné instrukce v jednom cyklu, je pro odlišení od skalárních strojů, které vydávají jednu instrukci v jednom cyklu.

V typickém superskalárním stroji operační kódy jsou ve vyvolaném instrukčním toku dekódovány a analyzovány dynamicky instrukční vydanou logikou za účelem určení zda instrukce lze provésti paralelně. Kriteria pro takové dynamické rozvrhování v poslední minutě jsou specifické pro každou architekturu instrukčního souboru, jakož i realizaci podléhající této architektuře v jakékoliv dané jednotce zpracovávající instrukce. Její účinnost je proto omezena složitostí logiky pro určení, které kombinace instrukcí lze provésti paralelně a doba cyklu jednotky pro zpracování instrukce se spíše zvětšuje. Zvětšené technické vybavení a doba cyklu těchto superskalárních strojů se stává dokonce velkým problémem v architekturách, které mají stovky různých instrukcí.

Při dynamickém rozvrhování, statickém rozvrhování nebo jejich kombinaci se vyskytují další nedostatky. Například je třeba přezkoumat každou skalární instrukci pokaždé znovu, jestliže je vyvolána k provedení rozhodnutí

zda je způsobilá pro paralelní provedení. K disposici není žádné opatření pro identifikaci a označení předem těch skalárních instrukcí, které jsou způsobilé pro paralelní provedení. Neexistuje žádné opatření pro identifikaci a včasné označení předem těch skalárních instrukcí, které jsou schopny paralelního provedení.

Jiným nedostatkem dynamického rozvrhování v realizaci superskalárních strojů je způsob jakým jsou skalární instrukce zkoušeny pro možné paralelní zpracování. Superskalární stroje zkoušejí skalární instrukce na základě popisů jejich operačního kódu a neexistuje opatření, aby byly vzaty v úvahu technické prostředky počítače. Instrukce jsou také vydávány metodou "první zařazen, první vybrán", čímž je eliminována možnost tvorby skupin, aby se zabránilo nebo omezil výskyt vzájemného blokování.

V několika dosavadních technických řešeních jsou vzaty v úvahu požadavky technických vybavení pro paralelní zpracování instrukcí. Jedním takovým řešením je stroj na zpracování velmi dlouhých instrukčních slov, ve kterých výkonný kompilárot přeřazuje instrukce tak, že je zjednodušeno rozvrhování instrukcí technickými prostředky. V tomto přiblížení musí být kompilátor složitější než jsou standardní kompilátory, takže je možno použít větší okénko pro vyhledání většího počtu paralelit v instrukčním toku. Výsledné instrukce nemohou být nutně cílovým kódem slučitelným s dosavadní stávající architekturou, čímž by byl jeden problém vyřešen, zatímco by vznikly další nové problémy. V podstatě vznikají také další problémy vzhledem k častému větvení, které omezuje jeho paralelitu.

Žádný z dosavadních postupů se proto nepřibližuje takovému paralelnímu zpracování, aby bylo dostatečně srozumitelné za účelem minimalizování všech možných vzájemných blokování, zatímco současně by bylo zabráněno novému

hlavnímu návrhu stavby instrukčního souboru a odstraněny složité logické obvody pro dynamické dekódování vyvolaných instrukcí.

To co je podle toho třeba je dokonalejší zpracování číselných dat, které usnadňuje paralelní provedení stávajících strojních instrukcí za účelem zvýšení účinnosti procesoru. Poněvadž počet instrukcí provedených za vteřinu je součinem základního časového cyklu procesoru a průměrného počtu požadovaných cyklů pro dokončení instrukce, je třeba nalézt takové řešení, které bere v úvahu oba tyto parametry. Mnohem specifičtěji je třeba mechanismu, který snižuje počet cyklů požadovaných k provedení instrukce pro danou architekturu. Kromě toho je třeba určitého zdokonalení, které by zredukovalo složitost technického vybavení nutného k podpoře paralelního provedení instrukcí, čímž je jakékoliv možné zvětšení časového cyklu zmenšeno na minimum. Nejvýše žádoucí je ještě pro navržené zdokonalení zajištění slučitelnosti realizace s již definovanou stávající architekturou, přičemž by byla zavedena paralelnost instrukční úrovně jak nového tak stávajícího strojového kódu.

Podstata vynálezu

Stručný obsah a předměty vynálezu

Se zřetelem ke shora uvedenému, je předmětem předloženého vynálezu vytvoření metody pro statickou analýzu, před dobou dekódování a provedení stávajících instrukcí, posloupnosti stávajících instrukcí pro generování složených instrukcí tvořených sousedními skupinami stávajících instrukcí, způsobilých k paralelnímu provedení. Souběžným předmětem je přidání příslušné řídicí informace do instrukčního toku včetně skupinové informace naznačující kde začíná složená instrukce, právě tak jako

oznámení počtu stávajících instrukcí, které jsou začleněny do každé složené instrukce.

Jiným předmětem vynálezu je analýza velkého okénka instrukčního slabikového toku před vyvoláním instrukce, přičemž okénko je nastavitelné do různých poloh v instrukčním slabikovém toku za účelem docílení optimálního selektivního seskupení jednotlivých sousedních instrukcí, které tvoří složenou instrukci.

Dalším předmětem vynálezu je opatření metody pro složení instrukcí o dříve zmíněných charakteristikách, která je aplikovatelná na architektury složitých instrukcí s proměnlivou délkou instrukcí a s daty smíšenými s instrukcemi, a která je rovněž aplikovatelná na architektury čítače přemístěných instrukcí, kde instrukce jsou obvykle konstantní délky a data nejsou smíšena s instrukcemi.

Dalším předmětem je opatření způsobu předběžného zpracování instrukčního toku pro vytvoření složených instrukcí, kde způsob lze realizovat programovým a/nebo technickým vybavením v různých bodech soustavy počítače před dekódováním a provedením instrukce. Souběžným předmětem je vytvoření metody předběžného zpracování stávajících instrukcí, která zpracovává tok binárních instrukcí jako část následného kompilátoru nebo část skladače vstupní paměti nebo jako část rychlé vyrovnávací paměti, jednotky pro složení instrukcí, a která může zahájit skládání instrukcí na začátku slabikového toku, aniž by byly známy meze instrukcí.

Vynález se tak vztahuje na metodu předběžného zpracování instrukčního toku pro vytvoření složených instrukcí sestavených ze skalárních instrukcí, které stále podržují svoje originální obsahy. Složené instrukce jsou tvořeny beze změny předmětového kódu skalárních instrukcí, které

tvoří složenou instrukci, čímž je umožněno stávajícím programům uskutečnit zlepšení výkonnosti strojů pro skladbu instrukcí, přičemž je zachována slučitelnost s předchozími realizovanými stroji skalárních instrukcí.

Mnohem specifičtěji vynálezem je vytvořen soubor pravidel pro skládání spočívající na analýze stávajících instrukcí za účelem jejich rozdělení do různých tříd. Analýza určuje, které instrukce se kvalifikují, buď s instrukcemi v jejich vlastních třídách nebo s instrukcemi v jiných třídách, pro paralelní provedení ve zvláštním uspořádání technických prostředků. Takováto pravidla pro skládání jsou považována za normu pro předběžné zpracování instrukčního toku za účelem vyhledání skupin dvou nebo více sousedních skalárních instrukcí, které lze provést paralelně. V některých případech určité druhy zablokovaných instrukcí lze skládat paralelním provedením, kde blokování je způsobilé sdružení ve speciální konfiguraci technických prostředků. V jiných uspořádáních kde blokování je nesdružitelné, instrukce, které mají blokování závislá na datech nebo závislá na technických prostředcích, jsou vyloučeny ze skupin tvořících složené instrukce.

Každá složená instrukce je identifikována řídicí informací jako je označení sdružené se složenou instrukcí, přičemž délka složené instrukce je způsobilá k převádění technických jednotek v širokém rozsahu, počínaje souborem dvou skalárních instrukcí až do jakéhokoliv maximálního počtu, který lze provést paralelně specifickou realizací technického vybavení. Poněvadž pravidla pro skládání jsou založena na identifikaci tříd instrukcí spíše než na jednotlivé instrukci, není žádná potřeba složitých matric ukazujících všechny možné kombinace specifických jednotlivých instrukcí. Zatímco jejich vlastní posloupnost zůstává nedotčena, jednotlivé instrukce jsou selektivně seskupeny a kombinovány s jednou nebo více ostatními sousedními

skalárními instrukcemi pro vytvoření složené instrukce, která obsahuje skalární instrukce, jež stále mají předmětový kód slučitelnosti s nesloženými skalárními instrukcemi. Řídicí informace je připojena pro identifikování informace týkající se provedení složených instrukcí.

Tyto a ostatní předměty, charakteristiky a výhody vynálezu jsou zřejmé specialistům z oboru se zřetelem k následujícímu podrobnému popisu a připojeným výkresům.

Přehled obrázků na výkresech

Obr. 1 základní diagram vyšší úrovně vynálezu.

Obr. 2 časový diagram realizace sekvenčního procesoru ukazující paralelní provedení určitých neblokovaných instrukcí, které byly selektivně seskupeny do toku složených instrukcí.

Obr. 3 časový diagram realizace multiprocesoru ukazující paralelní provedení skalárních a složených instrukcí, které nejsou vzájemně blokovány.

Obr. 4 /znázorněn jako obr. 4A a obr. 4B/ znázorňuje příklad možného selektivního roztřídění části instrukcí provedených stávajícím skalárním strojem.

Obr. 5 ukazuje typickou cestu programu vedenou od zdroje kódu ke skutečnému provedení.

Obr. 6 vývojový diagram ukazující generaci programu souboru složených instrukcí z programu v jazyce asembleru.

Obr. 7 vývojový diagram ukazující provedení programu souboru složených instrukcí.

Obr. 8 analytické schéma pro texty toku instrukcí s identifikovatelnými instrukčními referenčními body.

Obr. 9 analytické schéma pro texty toku instrukcí s proměnlivou délkou instrukcí bez referenčního bodu, ukazující jejich souběžné soubory možných složených identifikačních bitů.

Obr. 10 znázorňuje logickou realizaci schopnosti skládání instrukcí pro zpracování textu toku instrukcí podle obr. 9.

Obr. 11 vývojový diagram pro skládání toku instrukcí s referenčními značkami pro identifikaci instrukčních mezních referenčních bodů.

Obr. 12 ukazuje příklad řídicího pole složené instrukce.

Obr. 13 vývojové schéma pro odvození a použití pravidel pro skládání aplikovatelných na specifické uspořádání technických prostředků systému počítače a jeho zvláštní architektický instrukční soubor.

Obr. 14 ukazuje jak různá seskupení platných neblokovaných párů instrukcí tvoří vícenásobné složené instrukce pro sekvenční nebo větvené cílové provedení.

Obr. 15 ukazuje jak různá seskupení platných neblokovaných trojic instrukcí tvoří vícenásobné složené instrukce pro sekvenční nebo větvené cílové provedení.

Obr. 16 /znázorněn jako obr. 16A a obr. 16B/ je vývojové schéma pro složení instrukčního toku podobně jako v obr. 9, které obsahuje instrukce proměnlivé délky bez mezních referenčních bodů.

Obr. 17 schéma ukazující typické složitelné páry instrukčních kategorií pro část Systému/370 instrukční sady v obr. 4.

Příklad provedení vynálezu

Podstata předloženého vynálezu spočívá v předběžném zpracování souboru instrukcí nebo programu, které má provést počítač za účelem statického určení, které neblokované instrukce mají být kombinovány do složených instrukcí, přičemž je připojena řídicí informace pro identifikaci takových složených instrukcí. Takové určení je založeno na pravidlech pro skládání, která jsou vypracována pro soubor instrukcí zvláštní architektury. Stávající skalární instrukce jsou roztříděny na základě analýzy jejich operand, použití a funkce technického vybavení, takže seskupení instrukcí složením z důvodů vyloučení blokování vlivem nezpůsobilosti sdružování spočívá na porovnání kategorií instrukcí spíše než na porovnání specifických instrukcí.

Jak ukázáno v různých výkresech a dále podrobněji popsáno, vynález o názvu "Počítač pro převod jednotek složených instrukčních souborů" je určen pro tok skalárních instrukcí, které je třeba složit nebo seskupit dohromady před dobou dekódování instrukce tak, že jsou již označeny a identifikovány pro současné paralelní provedení příslušnými základními jednotkami. Poněvadž takové složení nemění cílový kód, mohou stávající programy docílit zlepšení výkonnosti, přičemž je dodržena slučitelnost s předchozími realizovanými systémy.

Jednotka 20 pro složení instrukce v obr. 1 přejímá tok 21 binárních skalárních instrukcí /včetně nebo bez dat v nich obsažených/ a selektivně seskupuje některé sousední skalární instrukce pro vytvoření kódovaných složených

instrukci. Výsledný instrukční tok 22 proto kombinuje skalární instrukce nezpůsobilé pro paralelní provedení a skládá instrukce tvořené skupinami skalárních instrukcí, které jsou způsobilé pro paralelní provedení. Když je nabídnuta skalární instrukce instrukční základní jednotce 24, je předána příslušné funkční jednotce pro sériové provedení. Když je složená instrukce nabídnuta instrukční základní jednotce 24, je každá z jejich skalárních složek předána příslušné funkční jednotce nebo sdružovací jednotce pro současné paralelní provedení. Typické funkční jednotky obsahují,ale nejsou omezeny na aritmetickou a logickou jednotku 26, 28, jednotku pro aritmetiku s pohyblivou čárkou 30 a jednotku pro generování paměťových adres 32. Příklad jednotky sdružující závislá data je popsán v souběžné přihlášce ser. No. 07/504,910 o názvu "Technické vybavení přístrojů sdružující závislá data" podané 4. 4. 1990.

Je třeba si uvědomit, že technika podle vynálezu je rozšiřována s úmyslem zjednodušit paralelní vydávání a provedení instrukcí ve všech architekturách počítačů, které zpracovávají v jednom cyklu vícenásobné instrukce /ačkoliv určité instrukce mohou vyžadovat více než jeden cyklus k provedení/.

Jak patrno z obr. 2 vynález lze realizovat v provedení sekvenčního procesoru, kde každá funkční provozní jednotka zpracovává skalární instrukci /S/ nebo alternativně složenou skalární instrukci /CS/. Jak patrno z výkresu instrukční tok 33 obsahující sekvenční skalární a složené skalární instrukce je opatřen řídicím označením /T/ přiřazeným každé složené instrukci. Tak první skalární 34 může být vykonána jedině funkční jednotkou A v cyklu 1; trojitě složená instrukce 36 identifikovaná označením T3 může mít své tři složené skalární instrukce vykonány paralelně funkčními jednotkami A, C a D v cyklu 2; jiná složená instrukce 38 identifikovaná označením T2 může mít svůj pár složených skalárních instrukcí vykonán paralelně funkčními jednotkami A a B v cyklu 3; druhá skalární instrukce 40 může být vykonána ojediněle funkční jednotkou C v cyklu 4; velká sku-

pina složených instrukcí 42 může mít své čtyři skalární instrukce vykonány paralelně funkčními jednotkami A-D v cyklu 5; a třetí skalární instrukce 44 lze vykonat jednotlivě funkční jednotkou A v cyklu 6.

Je důležité si uvědomit, že vícenásobné složení instrukce jsou způsobilé pro paralelní zpracování v určitých architekturách počítacích systémů. Například vynález lze potenciálně realizovat vybavenosti multiprocesoru jak ukazuje obr. 3, kde složená instrukce zpracovávaná jako jednotka pro paralelní vyvoláváni je jednou ze základních jednotek. Jak znázorněno na výkrese, tentýž instrukční tok 33 mohl by být zpracováván pouze ve dvou cyklech jak následuje. V prvním cyklu základní jednotka CPU # 1 provádí první skalární instrukci 34; funkční jednotky z CPU # 2 provedou trojitou složenou instrukci 36; a funkční jednotky z CPU # 3 provedou dvě složené skalární instrukce ve složené instrukci 38. Ve druhém cyklu základní jednotka CPU #1 provede druhou skalární instrukci 40; funkční jednotky z CPU # 2 provedou čtyři složené skalární instrukce ve složené instrukci 42; a funkční jednotka z CPU # 3 provede třetí skalární instrukci 44.

Příkladem architektury počítače, který je možno adaptovat na zpracování složených instrukcí je IBM systém/370 architektury úrovně instrukcí, ve kterém mohou být vydány vícenásobné skalární instrukce k vykonání v každém strojovém cyklu. V tomto pojetí strojový cyklus odpovídá všem zřetězeným krokům nebo stupňům potřebným k provedení skalární instrukce. Skalární instrukce pracuje s operandy o obsahu jednoznačných parametrů. Když je skalární tok složen, jsou selektivně seskupeny sousední skalární instrukce za účelem současného nebo paralelního provedení.

Soubory instrukcí pro různé architektury IBM systému/370 jako systém/370 s rozšířenou architekturou /370-XA/ a systém/370 architektury operačních systémů /370=ESA/

jsou dobře známy. Z tohoto hlediska doporučujeme publikaci "Principy činnosti IBM systému/370"/zveřejněno # GA22-7000-10 1987/, a "Principy činnosti, architektura operačních systémů IBM/370" /zveřejněno # SA22-7200-0 1988/.

Všeobecně zařízení pro skládání instrukce bude vyhledávat třídy instrukcí, které lze vykonávat paralelně a bude zajišťovat, že mezi členy složené instrukce nevznikne žádné vzájemné blokování, které by nebylo možno zvládnout technickými prostředky. Jsou-li nalezeny slučitelné posloupnosti instrukcí, je složená instrukce vytvořena.

Při podrobnější specifikaci je možno soubor instrukcí Systému/370 roztřídit na kategorie instrukcí, které mohou být provedeny paralelně ve speciální architektuře systému počítače. Instrukce v určitých z těchto kategorií lze kombinovat nebo skládat s instrukcemi téchže kategorií nebo s instrukcemi určitých jiných kategorií, aby byla vytvořena složená instrukce. Například část souboru instrukcí Systému/370 může být rozdělena do kategorií jak znázorněno v obr. 4. Rozumné zdůvodnění této kategorizace spočívá ve funkčních požadavcích instrukcí Systému/370 a v použití jeho technického vybavení pro typickou architekturu počítačového systému. Zbytek instrukcí Systému/370 není specificky uvažován pro skládání v tomto příkladovém spojení. To nezabraňuje jejich skládání pomocí metod a techniky předloženého a dále popsaného vynálezu. Je známo, že struktury technického vybavení potřebné k provádění složených instrukcí je možno běžně řídit horizontálními mikrokódy, které umožňují využít paralelnosti ve zbývajících instrukcích, které nebyly vybrány pro složení ani nebyly obsaženy v kategoriích podle obr. 4, čímž je zvýšena výkonnost.

Jednou z nejběžnějších posloupností v programech Systému/370 je provedení instrukce -test s maskou TM nebo srovnávací formáty /RX/ - srovnej C, srovnej polovinu CH, srovnej logicky CL, srovnej logicky ihned CLI, srovnej logicky test s maskou CLM, přičemž výsledku je použito k řízení vykonání instrukce typu - podmíněný skok BC /RX-formát/, podmíněný skok /RR-forma/, která ihned následuje. Výkonnost lze zlepšit paralelním provedením instrukcí - srovnej a skok - a to bývá někdy dynamicky uděláno vysoko výkonnými instrukčními procesory. Někdy vznikají obtíže v rychlém identifikování všech různých členů instrukcí třídy -srovnej- a všech členů instrukcí třídy -skok- v typické architektuře během dekódovacího procesu. Je to jeden z důvodů proč superskalární stroje obvykle berou v úvahu pouze malý počet specifických skalárních instrukcí pro možné paralelní zpracování. Takové omezené dynamické rozvrhování založené pouze na srovnání v poslední minutě dvou specifických instrukcí je naopak podle vynálezu odstraněno, protože analyzování všech členů tříd je předem splněné včas z důvodů dodržení přiměřených pravidel pro skládání při tvorbě složené instrukce, kterou je nutno vykonat.

Při dynamickém rozvrhování jednotlivých instrukcí vzniká velký problém po vyvolání programu jak patrno z obr. 4, který ukazuje, že dvoucestným složením padesátisedmi instrukcí je vytvořena matrice 57 x 57 více než tři tisíce možných kombinací. To je v ostrém protikladu s matricí 10 x 10 v obr. 17 pro tentýž počet instrukcí se zřetelem k možným kategoriím kombinací, které jsou k dispozici podle předloženého vynálezu.

Mnoho tříd instrukcí lze provést paralelně v závislosti na tom, jak je navrženo technické vybavení. Kromě shora popsaných složitelných párů - srovnávat a větvit -, je možno paralelně provést mnoho jiných složitelných kombinací

/viz obr. 17/, jako zavádění programu /kategorie 7/ složené s instrukcemi formátu RR /kategorie 1/, větvení /kategorie 3-5/ složené se zaváděcí adresou /kategorie 8/ apod.

V některých případech sekvenční příkaz ovlivní možnosti paralelního provedení a tím určí, zda lze složit dvě sousední instrukce. Se zřetelem k tomu řádkové hlavičky 45 identifikují kategorii první instrukce slabikového toku a sloupcové hlavičky 47 identifikují kategorii nejbližší instrukce, která následuje po první instrukci. Například -větvení /kategorie 3-5/ následována určitými -posuvy/kategorie 2/- jsou vždy složitelné 49, zatímco posuvy /kategorie 2/ následována - větvením /kategorie 3-5/ jsou jenom "někdy" složitelné 51.

Stav "někdy" identifikovaný jako "S" ve schématu na obr. 17 může být často změněn na "vždy" identifikovaný jako "A" ve schématu přidáním dalších funkčních jednotek technického vybavení v uspořádání soustavy počítače. Vezmeme-li v úvahu například sestavu, která zajišťuje dvoucestné složení a která nemá sdružovací jednotku pro sčítání-posouvání, ale místo toho má běžnou aritmetickou a logickou jednotku a samostatný přeřaďovač.Jinými slovy zde neexistuje vzájemné blokování sdružovacím technickým vybavením pro zpracování blokovaných instrukcí "sčítat-posun". Uvažujme následující instrukční sled:

```
AR R1, R2
SRL R3 po D2
```

Je jasné, že tento pár instrukcí je složitelný pro paralelní provedení. V některých případech však nebude složitelný vzhledem k neslučitelnému vzájemnému blokování, jak patrno z následujícího instrukčního sledu:

```
AR R1, R2
SRL R1 po D2
```

Tak schéma ukazuje, že kategorie 1 instrukce /AR/ následovaná kategorií 2 instrukce /SRL/ je někdy složitelná 53.

Při existenci aritmetické a logické jednotky, která sdružuje určitá vzájemná blokování, tak jako blokování sčítat/posunout, jak ukázáno shora, může ve schématu na obr. 17 "S" se změnit v "A". Podle toho pravidla pro složení musí být pohotová, aby reagovala na všechny změny provedené zvláštním uspořádáním soustavy počítače. Jako dodatečný příklad uvažujme instrukce obsažené v kategorii 1 složené s instrukcemi téže kategorie v následujícím instrukčním sledu:

AR R1, R2

SR R3, R4

Tento sled je bez vzájemných blokování náhodnými údaji a poskytuje následující výsledky, které obsahují dvě nezávislé instrukce Systému/370:

R1 = R1+R2

R3 = R3-R4

Provedení takového sledu by vyžadovalo dvě nezávislé a paralelní dvě až jednu aritmetickou a logickou jednotku, navrženou podle architektury instrukční úrovně. Tím se rozumí, že tyto dvě instrukce mohou být seskupeny do tvaru složené instrukce v uspořádání soustavy počítače, který má dvě takové aritmetické a logické jednotky. Tento příklad složení skalárních instrukcí je možno zevšeobecnit na všechny páry instrukčních posloupností, které jsou prosty blokování závislých dat a také blokování závislém na technickém vybavení.

V kterémkoli skutečném instrukčním procesoru existuje horní mez pro počet jednotlivých instrukcí, která může obsahovat složenou instrukci. Tato horní mez musí být specificky začleněna do technického a/nebo programové jednotky, která tvoří složené instrukce tak, že složené instrukce nebudou obsahovat více jednotlivých instrukcí /například párových skupin, trojitých skupin, skupin čtyř/ než je maximální způsobilost podléhající prováděcímu technickému vybavení. Tato horní mez je přísným důsledkem realizace technického vybavení ve zvláštní konfiguraci počítačové

soustavy a neomezuje ani celkový počet instrukcí, které mohou přijít v úvahu pro složení, ani délkou okénka skupiny v dané kódové posloupnosti, která má být analyzována pro složení.

Obecně platí, že čím větší délka okénka skupiny analyzované pro složení, tím větší paralelnost, kterou lze docílit vzhledem k výhodnějším složeným kombinacím. Z tohoto hlediska uvažujme posloupnost instrukcí v následující tabulce 1:

T a b u l k a 1

```
X1                ; jakákoliv složitelná instrukce
X2                ;      "          "          "
LOAD  R1, /X/     ; zaveď R1 z paměti umístění X
ADD   R3, R1      ; R3 = R3 + R1
SUB   R1, R2      ; R1 = R1 - R2
COMP  R1, R3      ; porovnej R1 s R3
X3                ; jakákoliv složitelná instrukce
X4                ;      "          "          "
```

Je-li horní mez pro složení vložená do technického vybavení rovna dvěma /nejvýše, dvě instrukce lze provést paralelně v jednom cyklu/, potom existuje více způsobů složení této posloupnosti instrukcí závislých na rozsahu programového vybavení pro skládání.

Kdyby rozsah složení byl roven čtyřem, potom programové vybavení pro složení by bralo v úvahu společně /X1, X2, LOAD, ADD/ a pak vysunulo dopředu jednu instrukci v době, aby byla vzata v úvahu dohromady /X2, LOAD, ADD, SUB/ a /LOAD, ADD, SUB, COMP/ a /ADD, SUB, COMP, X3/ a /SUB, COMP, X3, X4/, čímž je vytvořeno následující optimální párování jako uchazečů na složenou instrukci:

/--X1/ /X2 LOAD/ /ADD SUB/ /COMP X3/ /X4--/

Toto optimální párování podle vynálezu úplně odlehčuje vzájemná blokování mezi LOAD a ADD a mezi SUB a COMP, přičemž poskytuje dodatečné možnosti X1 složené s její předcházející instrukcí a X4 složené s její následující instrukcí.

Na druhé straně superskalární počítač, který dynamicky páruje instrukce do jeho instrukčního vydání logiky na základě dodržování metody "první zařazen, první vybrán", by vytvářel pouze následující párování jako kandidáty na paralelní provedení:

/X1 X2/ /LOAD ADD/ /SUB COMP/ /X3 X4/

Toto nepružné párování přivodí úplnou sanaci určitých blokujících instrukcí a docileno je pouze částečných úspěchů paralelního zpracování.

Vývojový diagram na obr. 13 podává vysvětlení různých kroků učiněných z důvodů rozhodnutí, které ze sousedních stávajících instrukcí ve slabikovém toku jsou kategorie nebo třídy, podle čehož jsou kvalifikovány k vytvoření společných skupin ve formě složené instrukce pro zvláštní uspořádání soustavy počítače.

Podle obr. 5 existuje mnoho možných míst v soustavě počítače, kde složení může nastat ať v programovém nebo technickém vybavení. Každé má svoje výhody a nevýhody. Jak v obr. 5 ukázáno, existují různé stupně, kterými program typicky prochází od zdroje kódu až po skutečné provedení. Během kompilační fáze je výchozí program přeložen do strojového kódu a uložen na disk 46. Během prováděcí fáze je program z disku 46 přečten a zaveden do hlavní paměti 48 speciálního uspořádání soustavy počítače 50, kde jsou instrukce provedeny příslušnými instrukčními základními jednotkami 52, 54, 56. Složení může nastat kdekoliv v průběhu této cesty. Obecně čím blíže ji je skladač umístěn

u instrukční základní jednotky nebo centrální základní jednotky CPU tím více se stává doba omezení naléhavější. Je-li skladač umístěn dále od centrální základní jednotky CPU, může být zkoušeno více instrukcí v rozměrnějším okénku instrukčního toku, aby se určilo nejlepší seskupení pro složení při vzrůstající pracovní výkonnosti. Takové brzké složení však přináší více než jeden dopad na zbytek systémové konstrukce podmiňující další vývoj a zvýšené náklady.

Jedním z důležitých předmětů vynálezu je opatření techniky pro stávající programy psané ve vyšších programovacích jazycích nebo stávajících asemblerových jazykových programech, aby byly zpracovány programovým vybavením, které může identifikovat posloupnosti sousedních instrukcí způsobilých k paralelnímu provedení jednotlivými funkčními jednotkami.

Vývojový diagram v obr. 6 ukazuje generování programu sady složených instrukcí z asemblerového jazykového programu v závislosti na sadě zakázkových skládacích pravidel 58, které nepříznivě se odrážejí jak na architektuře systému, tak technickém vybavení. Asemblerový jazykový program je připraven na vstup do programového skládacího zařízení 59, které tvoří program složených instrukcí. Následující bloky instrukcí o předem stanovené délce jsou analyzovány programovým skládacím zařízením 59. Délka každého bloku 60, 62, 64 ve slabikovém toku, která obsahuje skupinu instrukcí, jež mají být dohromady složeny, je závislá na složitosti zařízení pro skládání.

Jak ukazuje obr. 6 toto speciální skládací zařízení je navrženo pro dvoucestné složení instrukcí o počtu "m" a pevné délce v každém bloku. Primárním prvním krokem je stanovení zda první a druhá instrukce tvoří složitelný pár, potom druhá a třetí tvoří složitelný pár, potom třetí a čtvrtá tvoří složitelný pár a tak dále až do konce bloku.

Po identifikaci různých možných složitelných párů C1-C5 dalším velmi žádoucím krokem je určení optimálního výběru složených instrukcí tvarovaných sousedními skalárními instrukcemi pro paralelní provedení. Na příkladu je ukázáno možné složení instrukcí z následujících různých posloupností /nepředpokládá se větvení/: I1,C2,I3,I5,C3,C5,I10; I1,C2,I4,I5,I6,C4,I9,I10; C1,I3, I4,I5,C3,C5,I10; C1,I3,I4,I5,I6,C4,I9,I10. Při základním uspořádání speciálního technického vybavení může zařízení pro složení vybrat preferovanou posloupnost složených instrukcí a použít označení nebo identifikační bity pro identifikaci optimální posloupnosti složených instrukcí.

Neexistuje-li optimální posloupnost, mohou být identifikovány všechny složitelné sousední skalární instrukce tak, že větvení k cíli umístěné mezi různými složenými instrukcemi může využit kterýkoliv ze složených párů, se kterými se setká /viz obr. 14/. Kde jsou k dispozici vícenásobné skládací jednotky je možno současně složit vícenásobné následující bloky v toku instrukcí.

Specifická konstrukce programového skládacího zařízení nebude dále zde probírána, protože podrobnosti jsou specifické pro danou architekturu instrukčních sad a jsou základem realizace. Ačkoliv konstrukce takových složených programů je poněkud podobná koncepčně moderním kompilátorům, které vykonávají rozvrhování instrukcí a jiné optimalizace spočívající na specifických strojových architekturách, jsou použitá kriteria k dokončení takových složení podle tohoto vynálezu jednoznačná, jak nejlépe ukazuje vývojové schéma na obr. 13. V obou případech pro daný vstupní program a popis instrukční sady a také architektury technického vybavení /to je strukturální hlediska pro realizaci/ je vytvořen výstupní program. V případě moderního kompilátoru je na výstupu optimální nová posloupnost ze stávajících instrukcí. V případě vynálezu výstupem je série složených instrukcí,z nichž každá je tvarována skupinou

sousedních skalárních instrukcí způsobilých paralelního provedení se složenými instrukcemi smíchanými s nesložitelnými skalárními instrukcemi a spotřebnými řídicími bity k provedení složených instrukcí současně jako část výstupu.

Snadnější je samozřejmě předběžné zpracování instrukčního toku za účelem vytvoření složených instrukcí, jestliže existují již známé referenční body pro označení, kde instrukce začínají. Referenční bod jak je dále použito znamená nějaké označené pole nebo jinou indikaci, která poskytuje informaci o umístění instrukčních mezí. V mnohých počítačových systémech takový referenční bod je výhradně znám pouze z doby kompilace kompilátorem a jenom ze základní jednotky při vyvolávání instrukcí. Takový referenční bod není znám mezi dobou kompilace a vyvoláním instrukce, pokud je přijmuto speciální schéma referenčního značení.

Když je po době kompilace složení hotovo, kompilátor může označit referenčními značkami /viz obr. 11/, které slabiky obsahují první slabiku instrukce a které obsahují data. Tato zvláštní informace přispívá k mnohem účinnějšímu složení, ježto jsou známa přesná umístění instrukce. Samozřejmě kompilátor může identifikovat instrukce a rozlišovat mezi instrukcemi a daty jinými způsoby za účelem poskytnutí specifické informace skládacímu zařízení označením instrukčních mezí.

Je-li taková informace o instrukční mezi známa, pokračuje generování příslušných identifikačních bitů pro složení přímým postupem dopředu na základě pravidel pro složení stanovených pro speciální architekturu a uspořádání soustavy technického vybavení /viz obr. 8/. Není-li taková informace o instrukční mezi známa a instrukce jsou proměnlivé délky, nastává mnohem složitější problém /viz obr. 9 a 16/. Tyto obrázky náhodou spočívají na preferovaném kódovacím schématu, které je podrobněji popsáno

v tabulce 2A dole, kde při dvoucestném složení je k dispozici označení bitem "1", je-li instrukce složena s další instrukcí a označení bitem "0", není-li složena s další instrukcí.

Řídicí bity v řídicím poli přidané zařízením pro složení obsahují informaci, týkající se provedení složených instrukcí a mohou obsahovat tak malou nebo velkou informaci, jak je pokládáno za účelné pro zvláštní realizaci. V obr. 12 je uveden příklad osmibitového řídicího pole. V nejjednoduššim spojení v jeden celek je však žádán jen první řídicí bit pro označení začátku složené instrukce. Ostatní řídicí bity poskytují dodatečnou volitelnou informaci týkající se provedení instrukcí.

V alternativním kódovacím vzoru pro složené instrukce použitelném jak pro obě dvoucestná složení, tak pro složení velké skupiny, je stanoven první řídicí bit "1" pro označení, že odpovídající instrukce znamená začátek složené instrukce. Všechny ostatní členy složené instrukce budou mít svůj první řídicí bit stanoven "0". Nebude možno náhodou kombinovat danou instrukci s ostatními instrukcemi, takže taková daná instrukce se objeví jako složená instrukce o délce jedna. To znamená, že první řídicí bit bude stanoven "1", ale první řídicí bit následující instrukce bude stanoven také "1". Podle tohoto alternativního kódovacího schéma budou dekódovací technické prostředky schopny zjistit kolik instrukcí obsahuje složená instrukce kontrolou všech identifikačních bitů pro řady skalárních instrukcí spíše než toliko kontrolováním identifikačního bitu na začátku složené instrukce jako v preferovaném kódovacím schématu uvedeném dole v tabulkách 2A-2C.

Vývojový diagram v obr. 7 ukazuje typickou realizaci provedení programu sady složených instrukcí, který je generován technickými prostředky preprocesoru 66 nebo progra-

mového proprocesoru 67. Slabikový tok složených instrukcí vstupuje do rychlé vyrovnávací paměti 68 složených instrukcí, která slouží jako ukládací vyrovnávací paměť poskytující rychlý přístup ke složeným instrukcím. Výdej složené instrukce 69 vyvolá složené instrukce z rychlé vyrovnávací paměti 68 a vyšle její jednotlivé složené instrukce do příslušných funkčních jednotek k paralelnímu provedení.

Je třeba zdůraznit, že jednotky /Ci EU/ 71 k provedení složených instrukcí, takové jako aritmetické a logické jednotky ALUs v soustavě počítače složených instrukcí, jsou schopny provádět buď skalární instrukce jednu po druhé nebo alternativně složené skalární instrukce paralelně s ostatními složenými skalárními instrukcemi. Takové paralelní provedení může tedy být uděláno v různých typech prováděcích jednotek tak jako ALU's, jednotek 73 s pohyblivou řádovou čárkou /FP/, jednotek 75 pro generování paměťových adres /AU/ nebo vícenásobně v jednotkách téhož typu /FP1, FP2, atd./ v souladu s architekturou počítače a uspořádáním specifického systému počítače. Konfigurace technických prostředků, kterými lze realizovat předložený vynález, jsou tedy dostupné až do vlastně neomezeného počtu prováděcích jednotek z důvodu dosažení maximální výkonnosti při paralelním zpracování. Kombinováním několika stávajících instrukcí do jediné složené instrukce dovoluje jedné nebo více jednotkám pro zpracování instrukcí v soustavě počítače účinně dekódovat a provést paralelně tyto složené stávající instrukce bez zpoždění, které vzniká při běžném paralelním zpracování v soustavách počítače.

V nejjednodušších příkladových kódovacích schématech této přihlášky je přidána minimální informace ke složení do instrukčního toku jako jeden bit pro každé dvě slabiky textu /instrukce a data/. Ke každé instrukci ve složeném slabikovém toku lze obecně přidat označení obsahující řídicí informaci, což znamená ke každé nesložené skalární

instrukci právě tak jako ke každé složené skalární instrukci včetně páru, trojic, nebo větších složených skupin. Identifikační bity zde dále použité se týkají té části označení, které je specificky použito pro identifikaci a roztřídění těch složených skalárních instrukcí tvořících složenou skupinu ze zbytku nesložených skalárních instrukcí. Takové nesložené skalární instrukce zůstanou ve složeném instrukčním programu a když jsou vyvolány jsou provedeny jednotlivě.

V soustavě se všemi čtyřslabikovými instrukcemi srovnanými do čtyřslabikových mezí je přiřazeno jedno označení každému čtyřslabičnému textu. Podobně, mohou-li být srovnány instrukce libovolně, je zapotřebí označení každé slabiky textu.

Ve znázorněném spojení v jeden celek jsou všechny instrukce Systému/370 zarovnány do mezí na půlslova /dvě slabiky/ s délkou buď dvě nebo čtyři nebo šest slabik, přičemž pro každé půlslovo je třeba označení s jedním identifikačním bitem. Na příkladu malého seskupení složených párů sousedních instrukcí označuje identifikační bit "1", že instrukce začínající uvažovanou slabikou je složena s následující instrukcí, zatímco "0" naznačuje, že instrukce začínající uvažovanou slabikou není složena. Identifikační bit přiřazený půlslovům, které neobsahují první slabiku instrukce je ignorován. Identifikační bit pro první slabiku druhé instrukce ve složeném páru je také ignorován. /V některých situacích větvení však tyto identifikační bity nejsou ignorovány/. Výsledkem tohoto kódovacího postupu pro identifikační bity je, že v nejjednodušším případě dvoucestného složení, je třeba jenom jeden bit informace pro základní jednotku během činnosti pro identifikaci složené instrukce.

Kde je možno seskupit více než dvě skalární instrukce pro vytvoření složené instrukce, mohou být požadovány dodatečné identifikační bity pro opatření adekvátní řídicí informace. Z důvodů snížení počtu bitů požadovaných pro minimální řídicí informaci, existuje však stále jiný alternativní formát k pokračování ve sledování složené informace. Například je možno i při skládání velké skupiny docílit jeden bit pro instrukci při následujícím kódování: hodnota "1" znamená složení s následující instrukcí a hodnota "0" znamená nesložit s následující instrukcí. Složená instrukce tvořená skupinou čtyř jednotlivých instrukcí bude mít sled identifikačních bitů /1,1,1,0/ pro složení. Jako při provedení jiných zde popsaných složených instrukcí, jsou složené identifikační bity pro složení přidružené půlslovům, které nejsou instrukcemi a proto nemají žádné operační kódy, ignorovány v době provedení.

Podle preferovaného dále popsaného kódovacího schéma minimální počet identifikačních bitů potřebný pro poskytnutí další informace udávající specifický počet skutečně složených skalárních instrukcí je logaritmus při základu 2 /zaokrouhleno nahoru na nejbližší celé číslo/ maximálního počtu skalárních instrukcí, které lze seskupit k vytvoření složené instrukce. Například, je-li maximum dvě, pak je třeba jeden identifikační bit pro každou složenou instrukci. Je-li maximum tři nebo čtyři, potom je zapotřebí dvou identifikačních bitů pro každou složenou instrukci. Je-li maximum pět, šest, sedm nebo osm, potom je zapotřebí třech identifikačních bitů pro každou složenou instrukci. Toto kódovací schéma je uvedeno dále v tabulkách 2A, 2B, 2C:

Tabulka 2A /maximum dvě/

Identifikační bity	Zakódovaný význam	Celkem # složeno
0	Tato instrukce není složena s následující instrukcí	žádné
1	Tato instrukce je složena s jednou následující instrukcí	dvě

Tabulka 2B /maximum čtyři/

Identifikační bity	Zakódovaný význam	Celkem # složeno
00	Tato instrukce není složena s následující instrukcí	žádné
01	Tato instrukce je složena s jednou následující instrukcí	dvě
10	Tato instrukce je složena se dvěma následujícími instrukcemi	tři
11	Tato instrukce je složena se třemi následujícími instrukcemi	čtyři

Tabulka 2C /maximum osm/

Identifikační bity	Zakódovaný význam	Celkem složeno
000	Tato instrukce není složena s následující instrukcí	žádné
001	Tato instrukce je složena s následující instrukcí	dvě
010	Tato instrukce je složena se dvěma následujícími instrukcemi	tři
011	Tato instrukce je složena se třemi následujícími instrukcemi	čtyři
100	Tato instrukce je složena se čtyřmi následujícími instrukcemi	pět
101	Tato instrukce je složena s pěti následujícími instrukcemi	šest
110	Tato instrukce je složena se šesti následujícími instrukcemi	sedm
111	Tato instrukce je složena se sedmi následujícími instrukcemi	osm

Proto je z toho vyvozeno, že každé půlslovo potřebuje označení, ale podle tohoto preferovaného schéma základní jednotka ignoruje všechny až na označení první instrukce

v prováděném instrukčním toku. Jinými slovy je zkoušena slabika za účelem posouzení zda je to složená instrukce testováním, jejich identifikačních bitů. Není-li to začátek složené instrukce, jsou její identifikační bity nula. Je-li slabika začátkem složené instrukce obsahující dvě skalární instrukce, identifikační bity jsou "1" pro první instrukci a "0" pro druhou instrukci. Je-li slabika začátkem složené instrukce obsahující tři skalární instrukce, identifikační bity jsou "2" pro první instrukci a "1" pro druhou instrukci a "0" pro třetí instrukci. Jinými slovy identifikační bity identifikují pro každé půlslovo zda tato zvláštní slabika je nebo není začátkem složené instrukce, zatímco v téže době oznamují počet instrukcí, které tvoří složenou skupinu.

Tyto příkladové metody kódování složených instrukcí předpokládají, že tři instrukce jsou složeny k vytvoření trojité skupiny a druhá s třetí instrukcí jsou také složeny k vytvoření párové skupiny. Jinými slovy, jestliže se vyskytne v trojité skupině větvení k druhé instrukci, oznámí identifikační bit "1" pro druhou instrukci, že druhá a třetí instrukce budou provedeny jako složený pár paralelně, i kdyby první instrukce v trojité skupině nebyla provedena.

Vynález ovšem není omezen na uvedené speciální kódovací preferované schéma. Různá jiná kódovací pravidla tak jako dříve popsané alternativní kódovací schéma jsou možná v oblasti a podle výkladu vynálezu.

Odborníkům z tohoto oboru je zřejmé, že předložený vynález požaduje jenom jedno složení instrukčního toku pro speciální uspořádání počítačové soustavy, načež jakékoliv vyvolání složených instrukcí způsobí také vyvolání s tím slučitelných identifikačních bitů. Následkem toho odpadá neúčinné určování v poslední minutě a výběr určitých

skalárních instrukcí pro paralelní provedení, k čemuž dochází opakovaně při každém vyvolání stejných nebo odlišných instrukcí pro provedení v tak zvaném super skalárním stroji.

Vzdor všem výhodám skládání binárního instrukčního toku, stává se tento postup obtížným při určitých architekturách počítače pokud není vyvinuta technika k určování instrukčních mezí ve slabikovém řetězci. Jsou-li povoleny proměnlivé délky instrukcí, je takové určování složité a ještě složitější při vzájemném smíšení dat a instrukcí nebo je-li dovoleno upravovat přímo instrukční tok. V době provedení musí ovšem instrukční meze být známy s ohledem na správné provedení. Jakmile je však složení dostatečně brzo hotovo před provedením instrukce, byla vyvinuta jednoznačná technika pro složení instrukcí bez znalosti, kde instrukce začínají a bez znalosti, které slabiky jsou data. Tato technika je dále povšechně popsána a lze ji použít k tvorbě složených instrukcí vytvořených ze sousedních párů skalárních instrukcí právě tak jako k tvorbě složených instrukcí vytvořených z větších skupin skalárních instrukcí. Tato technika je použitelná pro všechny instrukční soubory různých běžných typů architektur, včetně čítače přemístěných instrukcí RISC, v jehož architekturách instrukce jsou obvykle konstantní délky a nejsou smíšeny s daty. Další podrobnosti této techniky skládání jsou popsány v souběžné přihlášce ser. No. 07/519,382 s názvem "Universální technika pro složení instrukčních úrovní paralelních procesorů" podaná 4. 5. 1990.

Všeobecně vzato, technika složení je k dispozici pro složení dvou nebo více skalárních instrukcí z instrukčního toku bez znalosti bodu rozběhu nebo délky každé jednotlivé instrukce. Typické instrukce již obsahují operační kód v předem určeném místě pole, které identifikuje instrukci a její délku. Ty sousední instrukce, které se kvalifikují pro paralelní provedení ve speciální konfiguraci počítačové

soustavy jsou opatřeny příslušnými značkami pro oznámení, že jsou uchazeči na složení. V IBM Systému/370, v jehož architektuře jsou instrukce buď dvě, čtyři nebo šest slabik dlouhé, předpokládá se, že polohy pole operačního kódu jsou založeny na odhadnuté instrukční délce kódu. Hodnota každého označení založená na předpokládaném operačním kódu je zaznamenána a instrukční délka kódu v předpokládaném operačním kódu je použita k umístění celé posloupnosti možných instrukcí. Jakmile jsou stanoveny instrukční meze, jsou použity odpovídající správné hodnoty označení k identifikaci začátku složené instrukce a ostatní nesprávně generovaná označení jsou ignorována.

Tato jednoznačná technika složení je znázorněna jako příklad na výkresech obr. 8-9 a 14-15, kde jsou definována pravidla složení určující, že všechny instrukce o délce dvou slabik nebo čtyř slabik jsou jedna s druhou složitelné /to znamená, že dvouslabičná instrukce je schopná paralelního provedení v této speciální konfiguraci počítače s jinou dvouslabičnou nebo jinou čtyřslabičnou instrukcí/. Příkladná pravidla pro složení dále ukládají, že všechny instrukce dlouhé šest slabik nejsou vůbec složitelná /to znamená, že v této speciální konfiguraci počítače šestislabičná instrukce je způsobilá sama o sobě jediného provedení/. Vynález není samozřejmě omezen na tato pravidla pro příkladné složení, ale je použitelný pro jakýkoliv soubor pravidel na složení, která definují kritéria pro paralelní provedení stávajících instrukcí ve speciální konfiguraci pro danou architekturu počítače.

Sada instrukcí použitá jako příklad složení v těchto technikách podle vynálezu je brána z architektury Systému/370. Přezkoušením operačního kódu každé instrukce lze určit typ a délku každé instrukce a potom je generováno řídicí označení obsahující identifikační bity pro tuto specifickou instrukci jak dále podrobněji popsáno. Před-

ložený vynález není ovšem omezen na jakoukoliv specifickou architekturu nebo instrukční soubor a shora zmíněná pravidla jsou jenom příležitostným příkladem.

Preferované kódovací schéma složených instrukcí bylo již znázorněno dříve v celkovém přehledu na tabulkách 2A - 2C.

V prvém případě s pevnou délkou instrukcí bez smíšených dat a se známým umístěním referenčního bodu operačního kódu, může složení pokračovat v souladu s aplikovatelnými pravidly pro tuto speciální konfiguraci počítače. Poněvadž pole reservované pro operační kód také obsahuje délku instrukce je běžně určena posloupnost skalárních instrukcí, přičemž každá instrukce posloupnosti může se považovat za uchazeče pro paralelní provedení s následující instrukcí. První kódovaná hodnota v řídicím označení oznamuje, že instrukce není složitelná s další instrukcí, zatímco druhá kódovaná hodnota řídicího označení oznamuje, že instrukce je složitelná pro paralelní provedení s další instrukcí.

V druhém případě instrukcí s proměnlivou délkou bez smíšených dat a se známým umístěním referenčního bodu pro operační kód a také pro kód délky instrukce /který je v Systému/370 obsažen jako část operačního kódu/, složení může postupovat běžným způsobem. Operační kódy jak patrno z obr. 8 oznamují instrukční posloupnost 70 jak následuje: první instrukce je dlouhá 6 slabik, druhá a třetí každá délky 2 slabik, čtvrtá je dlouhá 4 slabiky, pátá je dlouhá 2 slabiky, šestá je dlouhá 6 slabik a sedmá a osmá každá délky 2 slabik.

C-vektor 72 v obr. 8 ukazuje hodnoty identifikačních bitů /na výkresech zvaných bity složení/ pro zvláštní posloupnost 70 instrukcí kde referenční bod oznamuje, že je

znám start první instrukce. Na základě hodnot těchto identifikačních bitů druhá a třetí instrukce tvoří složený pár jak označuje "1" v identifikačním bitu pro druhou instrukci. Čtvrtá a pátá instrukce tvoří jiný složený pár jak označuje "1" v identifikačním bitu čtvrté instrukce. Sedmá a osmá instrukce také tvoří složený pár jak označuje "1" identifikačního bitu pro sedmou instrukci.

C-vektor 72 v obr. 8 je poměrně snadno generován, neboť neobsahuje žádné datové slabiky smíšené s instrukčními slabikami a instrukce jsou všechny stejné délky o známých mezích.

Jiná situace nastává ve třetím případě, kde instrukce jsou smíšeny s neinstrukcemi a referenční bod poskytuje stále oznámení začátku instrukce. Základní diagram v obr. 11 ukazuje jeden způsob značení instrukčního referenčního bodu, kde každé půlslovo bylo označeno referenčním označením pro indikaci, že obsahuje první slabiku instrukce nebo ji neobsahuje. Může tomu tak být jak u instrukcí s pevnou tak proměnlivou délkou. Stanovením referenčního bodu je nezbytné ohodnotit část dat slabikového toku pro možné složení. Jednotka pro složení podle toho může přeskočit a ignorovat všechny slabiky neinstrukční.

Mnohem složitější situace vzniká kde slabikový tok obsahuje instrukce proměnlivé délky /bez dat/, není však známo, kde začíná první instrukce. Poněvadž maximální délka instrukce je šest slabik a poněvadž instrukce jsou seřazeny do dvouslabikových mezí, existují tři možné startovací body pro první instrukci toku. Technika podle toho dává na vybranou všechny možné startovací body pro první instrukci v textu slabikového toku 79 jak patrno z obr. 9.

Posloupnost 1 předpokládá, že první instrukce zahajuje první slabikou a postupuje se složením dopředu podle

předpokladu. V tomto příkladovém spojení v jeden celek délka pole je také rozhodující pro hodnotu C-vektoru každé možné instrukce. Proto C-vektor 74 má jenom pro posloupnost 1 hodnotu "1" pro první instrukci možného složeného páru tvořenou kombinacemi 2-slabikových a 4-slabikových instrukcí.

Posloupnost 2 předpokládá, že první instrukce zahajuje třetí slabikou /začátek druhého půlslova/ a postupuje dopředu podle tohoto předpokladu. Hodnota délky pole pro třetí slabiku je 2 oznamující, že další instrukce začíná pátou slabikou. Procházením vpřed každou možnou instrukcí založenou na délce pole předchozí instrukce, je celý potenciál instrukcí posloupnosti 2 generován podle možných identifikačních bitů jak patrno z C-vektoru 76.

Posloupnost 3 předpokládá, že první instrukce začíná pátou slabikou /počátek třetího půlslova/ a pokračuje jak předesláno. Hodnota v délce pole pro pátou slabiku je 4 znamenající, že příští instrukce začíná devátou slabikou. Pokračováním vpřed každou možnou instrukcí založenou na hodnotě délky pole předchozí instrukce, celý potenciál instrukcí posloupnosti 3 je generován podle možných identifikačních bitů jak ukazuje C-vektor 78.

V některých případech uvedeno tři posloupnosti potenciálních instrukcí se sbíhají do jediné posloupnosti. V obr. 9 je věnována pozornost uvedeným třem posloupnostem, které se sbíhají do mezí instrukce osmé slabiky na konci 80. Také je poznamenáno, že začínají-li další posloupnosti na konci šesté, osmé a desáté slabiky, sbíhají se také rychle. Posloupnosti 2 a 3 při sbíhání do mezí instrukce na konci 82 čtvrté slabiky jsou mimo fázi při skládání až do konce šestnácté slabiky. Jinými slovy tyto dvě posloupnosti berou v úvahu různé páry instrukcí založené na stejné posloupnosti instrukcí. Poněvadž sedmnáctá slabika za-

činá nesložitelnou instrukci v 84, je mimo fázová sbíhavost ukončena.

Nevznikne-li platná sbíhavost, je třeba, aby všechny tři možné posloupnosti instrukcí pokračovaly do konce okénka. Kde však dojde k platné sbíhavosti a je zjištěna, sdruží se počet posloupností ze tří do dvou /jedna z identických posloupností se stane nezpůsobilá provozu/ a v některých případech ze dvou do jedné.

Před sbíhavostí jsou tedy určeny pokusné meze instrukce pro každou možnou instrukční posloupnost a přiděleny identifikační bity pro každou takovou instrukci označující umístění potenciálních složených instrukcí. Z obr. 9 je zřejmé, že tato technika generuje tři samostatné identifikační bity pro každé dvě slabiky textu. Za účelem vytvoření shody s předběžným zpracováním v případech A-D, je žádoucí zredukování tří možných posloupností do jediné posloupnosti s identifikačními bity, kde je přidělen pouze jeden bit každému půlslovu. Poněvadž je zapotřebí jen jediná informace zda je běžná instrukce složena s následující instrukcí, mohou být tři bity logicky čteny jako "0" k vytvoření jediné posloupnosti v CC-vektoru 86.

Pro účely paralelního provedení jsou sdružené identifikační bity sdruženého CC-vektoru rovnocenné jednotlivým C-vektorům jednotlivých tří posloupností 1-3. Jinými slovy sdružené identifikační bity CC-vektoru umožňují kterékoliv možné posloupnosti správné paralelní provedení složených instrukcí nebo jednotlivých nesložených instrukcí. Sdružené identifikační bity také přispívají ke správnému větvení. Vznikne-li například větev na začátku deváté slabiky 88, potom devátá slabika musí zahájit instrukci. Jinak v programu je chyba. Identifikační bit "1" přidružený k deváté slabice je použit ke korekci paralelního provedení takové instrukce s její příští postupující instrukcí.

Různé kroky v metodě pro složení znázorněné na obr. 9 jak shora popsáno jsou zobrazeny v obr. 16, který sám o sobě přispívá k vysvětlení.

Nejlepší dobou pro poskytnutí informace referenčním bodem pro instrukční meze je doba kompilace. Jak patrno z obr. 11 referenční označení 101 mohlo by být připojeno v době kompilace k identifikaci zahájení každé instrukce. Tím je umožněno zařízení pro složení postupovat zjednodušenou technikou ve shora zmíněném prvním, druhém a třetím případě jak již dříve uvedeno. Kompilátor by ovšem mohl identifikovat instrukční meze a rozlišovat mezi instrukcemi a daty jinými způsoby z důvodu zjednodušení práce jednotky pro složení a vyhnutí se komplikacím techniky takové jak znázorněno na obr. 9.

Obr. 10 znázorňuje vývojový diagram možné realizace skládající jednotky pro zpracování instrukčního toku, jak patrno z obr. 9. Vícenásobný počet skládacích jednotek 104, 106, 108 by mohl být z důvodu účinnosti tak velký jako je počet půlslov v textu vyrovnávací paměti. Tři skládající jednotky v této verzi by začínaly zpracovávat posloupnosti příslušné první, třetí a páté slabiky. Po ukončení možné instrukční posloupnosti, každá skládající jednotka zahájí zkoušení dalšího možného sledu přeloženého do šesti slabik z jejího předchozího sledu. Každá skládající jednotka tvoří složené identifikační bity /hodnoty C-vektoru/ pro každé půlslovo textu. Tři posloupnosti ze tří skládajících jednotek jsou přečteny 110 a výsledné sdružené identifikační bity /hodnoty CC-vektoru/ jsou uloženy do paměti a přiřazeny jejich odpovídajícím textovým slabikám.

Prospěšnou výhodou poskytovanou sdruženými identifikačními bity CC-vektoru je tvoření sledu vícenásobně platných složených bitů založených na tom, která instrukce je adresována cílem větvení. Jak nejlépe patrno z obr. 14-15

je možno z téhož slabikového toku složené instrukce různě tvarovat.

Obr. 14 znázorňuje možné kombinace složených instrukcí, když konfigurace počítače poskytuje paralelní vydání a provedení ne více než dvou instrukcí. Kde je instrukční tok 90 se složenými instrukcemi zpracován do jednoho normálního sledu, bude složená instrukce I vydána k paralelnímu provedení založenému na dekódování identifikačního bitu první slabiky CC-vektoru 92. Vyskytne-li se však větev k páté slabice, bude vydána složená instrukce II k paralelnímu provedení založená na dekódování identifikačního bitu pro pátou slabiku.

Podobně normální sekvenční zpracování jiného složeného slabikového toku 94 má za následek sekvenčně prováděné složené instrukce IV, VI a VIII /instrukce složek v každé složené instrukci jsou přitom prováděny paralelně/. Větvení ve třetí slabice složeného slabikového toku má za následek sekvenční provádění složených instrukcí V a VII, a instrukce začínající patnáctou slabikou /tvoří druhou část složené instrukce VIII/ bude vydána a provedena jednotlivě, přičemž vše spočívá na identifikačních bitech v CC-vektoru 96.

Větvení v sedmé slabice má za následek sekvenční provádění ve složených instrukcích VI a VIII, a větvení v jedenácté slabice má za následek sekvenční větvení ve složené instrukci VIII. Větvení v deváté slabice naopak má za následek sekvenční provádění ve složené instrukci VII /je tvarována druhou částí složené instrukce VI a první části složené instrukce VIII/.

Tak jsou zanedbány identifikační bity "1" v CC-vektoru 96 pro složené instrukce IV, VI a VIII, když je prováděna některá ze složených instrukcí V nebo VII. Alternativně

jsou zanedbány identifikační bity "1" v CC-vektoru 96 pro složené instrukce V a VII, když je prováděna kterákoliv ze složených instrukcí IV, VI nebo VIII.

V obr. 15 jsou znázorněny možné kombinace složených instrukcí, když uspořádání počítače poskytuje paralelní vydávání a provádění až do tří instrukcí. Kde instrukční tok 98 obsahující složené instrukce je zpracováván do normální posloupnosti, budou provedeny složené instrukce X /trojitá skupina/ a XIII /párová skupina/. Větvení v jedenácté slabice naopak má za následek provedení složené instrukce XI /trojitá skupina/, a větvení ve třinácté slabice má za následek provedení složené instrukce XII /odlišná trojitá skupina/.

Bity "2" identifikátoru v CC-vektoru 99 pro složené instrukce XI a XII jsou zanedbány, když jsou provedeny složené instrukce X a XIII. Na druhé straně, když je provedena složená instrukce XI, jsou zanedbány bity identifikátoru pro ostatní tři složené instrukce X, XII, XII. Podobně, když je provedena složená instrukce XII, jsou zanedbány bity identifikátoru pro ostatní tři složeno instrukce X, XI, XIII.

Existuje mnoho možných konstrukcí jednotky pro složení instrukce závisející na jejím umístění a znalosti textových obsahů. V nejjednodušší situaci bylo by žádoucí, aby kompilátor ukazoval pomocí označení, které slabiky obsahují první slabiku instrukce a které obsahují data. Tato zvláštní informace má za následek mnohem účinnější jednotku pro složení, neboť jsou známa přesná umístění instrukce /viz obr. 13/. Znamená to, že složení lze vždy provádět jako v situacích případu C za účelem generování C-vektoru identifikačních bitů pro každou složenou instrukci /viz obr. 8/. Kompilátor mohl by tak přidat jinou informaci jako předpověď statického větvení nebo dokonce směrnice do jednotky pro složení.

Pro rozlišení dat od instrukcí lze použít jiných postupů kde instrukční tok má být složen je uložen do paměti. Například jsou-li části dat řídké, vyžadoval by jednoduchý seznam adres obsahující data méně místa než označení. Takové kombinace technických a programových prostředků pro složení skýtají mnoho variant pro účinné vytváření složených instrukcí.

Zatímco byla popsána příkladná preferovaná spojení v jeden celek podle vynálezu, bude na odbornících v této oblasti techniky, aby posoudili různé modifikace a změny, které lze odvodit z myšlenky a obsahu vynálezu jak je definován v následujících nárocích.

PATENTOVÉ NÁROKY

1. ~~Metoda~~ Způsob zpracování sledu instrukcí ~~k docílení paralelního provedení stávajících instrukcí ve speciální konfiguraci systému ke zpracování dat,~~ vyznačující se tím, že obsahuje přidělování určitých stávajících instrukcí do vícenásobných kategorií, porovnání kategorií sousedních stávajících instrukcí v instrukčním toku k rozhodnutí zda jsou sousední stávající instrukce způsobilé paralelního provedení ve speciální konfiguraci systému ke zpracování dat a identifikování pomocí ukazatele složení těch sousedních stávajících instrukcí, které jsou pomocí porovnávacích kroků uznány za způsobilé paralelního provedení.

2. ~~Metoda~~ Způsob podle nároku 1, vyznačující se tím, že přidělovací a porovnávací kroky jsou vykonány dříve, než jsou stávající instrukce vyvolány k provedení.

3. ~~Metoda~~ Způsob podle nároku 1 nebo 2, vyznačující se tím, že přidělovací krok obsahuje přidělení určitých instrukcí do nejméně jedné kategorie tak, že zmíněný porovnávací krok přiměje identifikační krok k identifikaci nejméně dvou sousedních stávajících instrukcí pomocí indikátoru složení u nejméně jedné kategorie pro paralelní provedení.

4. ~~Metoda~~ Způsob podle nároku 1 až 3, vyznačující se tím, že přidělovací krok obsahuje přidělení určitých instrukcí do nejméně dvou různých nepřesahujících kategorií tak, že zmíněný porovnávací krok přiměje identifikační krok k identifikaci indikátorem složení nejméně dvou sousedních stávajících instrukcí ze zmíněných dvou kategorií, navzájem pro paralelní provedení.

5. ~~Metoda~~ Způsob podle nároků 1 až 4, vyznačující se tím, že identifikační krok obsahuje upuštění od cílového kódu stávajících instrukcí v jeho původním tvaru pro jednotlivé provedení nebo pro paralelní provedení s jinou instrukcí.

6. ~~Metoda~~ Způsob podle nároků 1 až 5, vyznačující se tím, že přidělovací krok bere v úvahu vzájemné blokování závislých dat mezi instrukcemi právě tak jako existenci relativního blokování sdružených funkčních jednotek ve speciální konfiguraci systému zpracovávajícího data.

7. ~~Metoda~~ Způsob podle jednoho z nároků 1 až 6, vyznačující se tím, že přidělovací krok bere v úvahu vzájemné blokování mezi instrukcemi závislé na technických prostředcích právě tak jako existenci relativního blokování sdružených funkčních jednotek ve speciální konfiguraci systému pro zpracování dat.

8. ~~Metoda~~ Způsob podle jednoho z nároků 1 až 7, vyznačující se tím, že porovnávací krok obsahuje porovnávání první stávající instrukce s druhou sousední následující instrukcí pro možné složení s každou další, a potom porovnávání druhé instrukce s třetí sousední následující instrukcí pro možné složení s každou další, k identifikování vícenásobných složených instrukcí identifikovaných příslušnými indikátory pro vícenásobné složení.

9. ~~Metoda~~ Způsob podle nároku 8, vyznačující se tím, že obsahuje dodatečný krok pro určení optimální posloupnosti vícenásobných složených instrukcí pro danou část instrukčního toku.

10. ~~Metoda~~ Způsob podle jednoho z nároků 1 až 9, vyznačující se tím, že porovnávací krok k určení zda sousední stávající instrukce jsou způsobilé paralelního provedení je založen na použití technických prostředků spíše než popisu operačního kódu.

11. ~~Systém~~ Zařízení pro zpracování posloupnosti instrukcí odebraných ze stávajícího sledu strojových instrukcí za účelem přípravy posloupnosti pro paralelní provedení speciál-

ní konfigurací počítače, vyznačující se tím, že obsahuje soubor pravidel spočívající na speciální konfiguraci počítače indikující určité strojové instrukce, které jsou způsobilé k paralelnímu provedení s jinými strojovými instrukcemi a prostředky složení pro předběžné zpracování před vyvoláním binárního instrukčního toku s obsahem stávajících instrukcí, přičemž zmíněné prostředky složení mají přístup k souboru pravidel pro identifikaci sousedních instrukcí způsobilých paralelního provedení ve speciální konfiguraci počítače a pro vytvoření složené posloupnosti instrukcí mající řídicí pole k identifikování jednotlivých sousedních instrukcí způsobilých pro paralelní provedení a které dohromady tvoří složenou instrukci.

12. ~~Systém~~ Zařízení podle nároku 11, vyznačující se tím, že prostředkem ke složení je program předběžného zpracování programovými prostředky použitými jako část kompilátoru nebo následného kompilátoru.

13. ~~Systém~~ Zařízení podle nároku 11 nebo 12, vyznačující se tím, že prostředek složení představuje jednotka pro předběžné zpracování technickými prostředky, která zpracovává instrukce před jejich vyvoláním z rychlé vyrovnávací paměti.

14. ~~Systém~~ Zařízení podle jednoho z nároků 11 až 13, vyznačující se tím, že prostředek složení je způsobilý vytvářet složené posloupnosti instrukcí s řídicím polem k identifikování párů sousedních instrukcí způsobilých pro paralelní provedení.

15. ~~Systém~~ Zařízení podle bodu 14, vyznačující se tím, že prostředek složení je způsobilý tvořit složené posloupnosti instrukcí s řídicím polem pro identifikování skupin tří nebo více sousedních instrukcí způsobilých paralelního provedení.

16. ~~Systém~~ Zařízení podle jednoho z nároků 11 až 15, vyznačující se tím, že řečené řídicí pole obsahuje nejméně jeden identifikační bit sdružený s každou složenou instrukcí.

17. ~~Systém~~ Zařízení podle nároku 16, vyznačující se tím, že řečené řídicí pole obsahuje identifikační část bitu a řídicí část bitu odlišnou od řečené identifikační části bitu.

18. ~~Systém~~ Zařízení podle jednoho z nároků 11 až 17, vyznačující se tím, že řečené řídicí pole obsahuje nejméně jeden identifikační bit přidružený ke každé jednotlivé instrukci tvořící složenou instrukci.

19. ~~Systém~~ Zařízení podle nároku 18, vyznačující se tím, že řídicí pole obsahuje nejméně jeden identifikační bit přidružený ke každé jednotlivé instrukci netvořící složenou instrukci.

20. ~~Systém~~ Zařízení podle jednoho z nároků 11 až 19, vyznačující se tím, že zmíněný soubor pravidel je založen na použití technických prostředků spíše než na popisu operačního kódu.

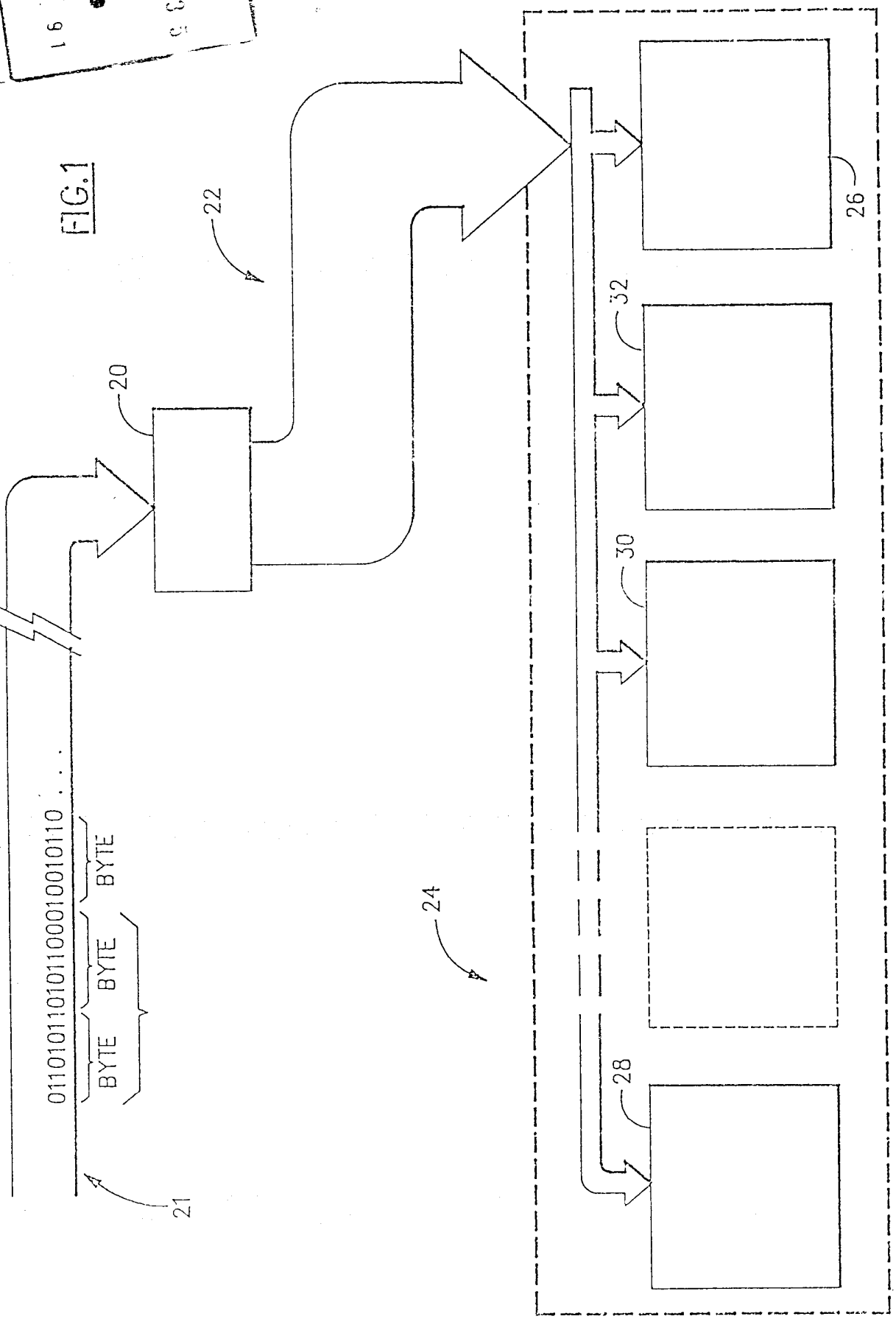
21. ~~Systém~~ Zařízení pro generování instrukčního toku pro paralelní provedení ve speciální konfiguraci počítače, vyznačující se tím, že obsahuje seznam stávajících skalárních instrukcí seskupených do vícenásobných kategorií založený na způsobilosti stejných nebo různých kategorií instrukcí k paralelnímu provedení ve speciální konfiguraci počítače, prostředky pro předběžné zpracování k přijmu na vstupu skalárních instrukcí jako části binárního slabikového toku a pro rozhodnutí podle zmíněného seznamu, které sousední skalární instrukce jsou kandidáty pro paralelní provedení ve speciální konfiguraci počítače, a prostředky ke složení pro tvorbu označení přidružených určitým skalárním

instrukcím v instrukčním toku indikujícím, které sousední skalární instrukce jsou částí složené instrukce způsobilé pro paralelní provedení, a které skalární instrukce v instrukčním toku nejsou způsobilé pro paralelní provedení ve speciální konfiguraci počítače.

22. ~~Systém~~ Zařízení podle nároku 21, vyznačující se tím, že obsahuje dále řídicí prostředky pro kontrolování instrukčního toku a pro vydání vícenásobných složených instrukcí pro paralelní provedení s každou jinou.

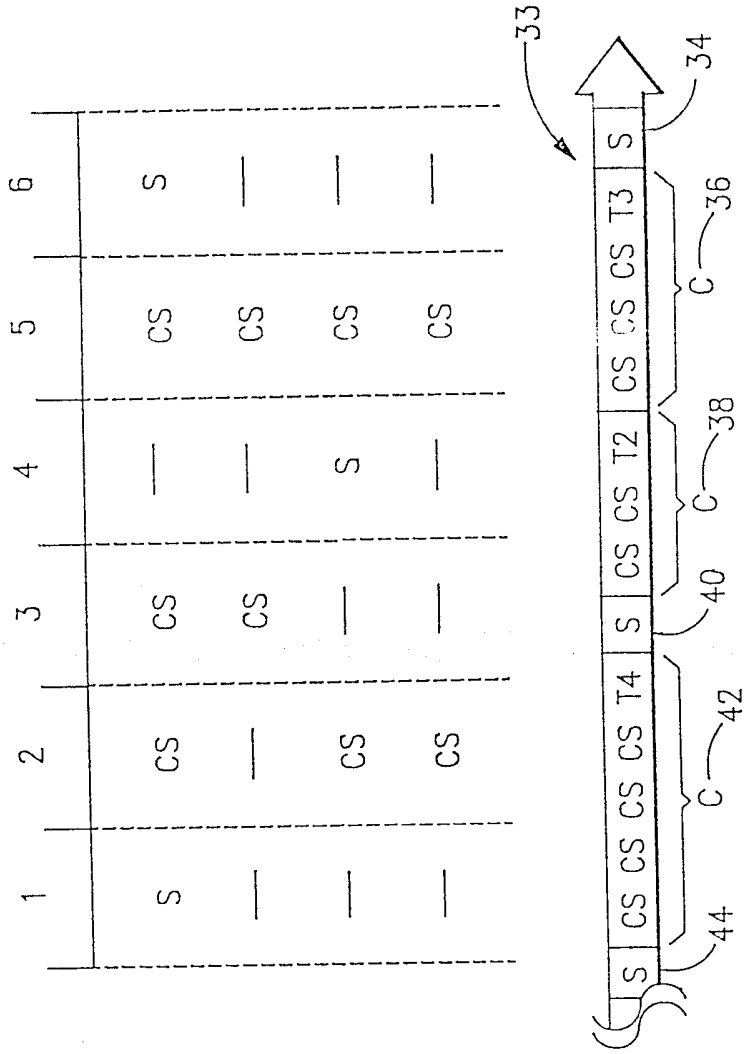
PV 936-91

FIG.1



PV 936-91G

FIG.2



CPU #1	FU-A		
	FU-B		
	FU-C		
	FU-D	S	S
CPU #2	FU-A	CS	CS
	FU-B	CS	CS
	FU-C	CS	CS
	FU-D	—	CS
CPU #3	FU-A	CS	—
	FU-B	CS	S
	FU-C	—	—
	FU-D	—	—

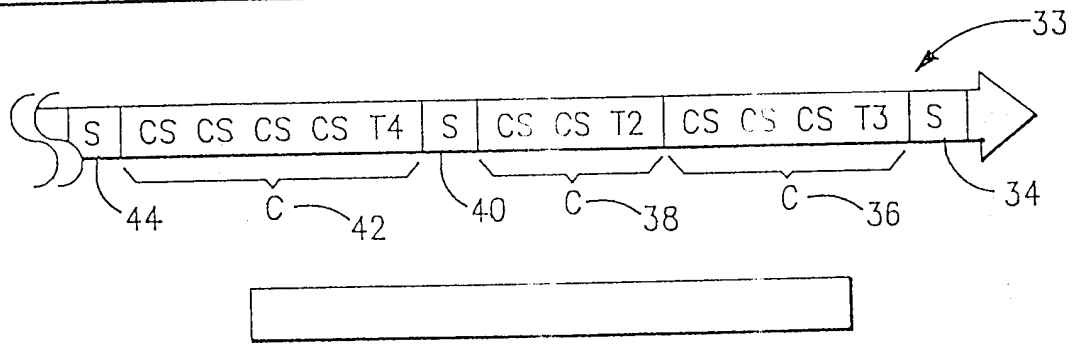
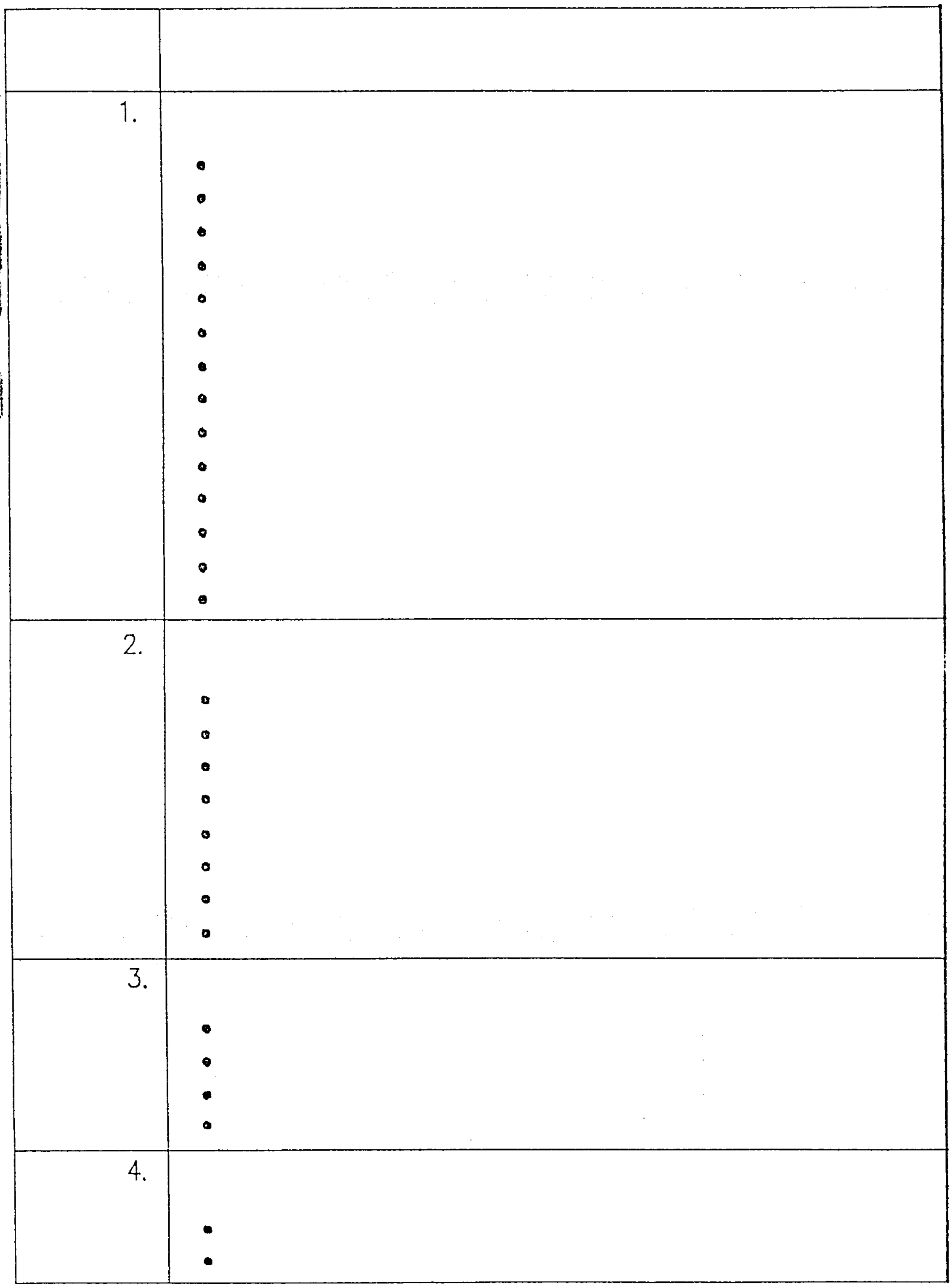


FIG.3

PV 936-91G



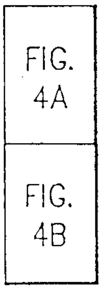


FIG.4

FIG.4A

5.	•
6.	•
7.	•
8.	
9.	•
10.	

FIG.4B

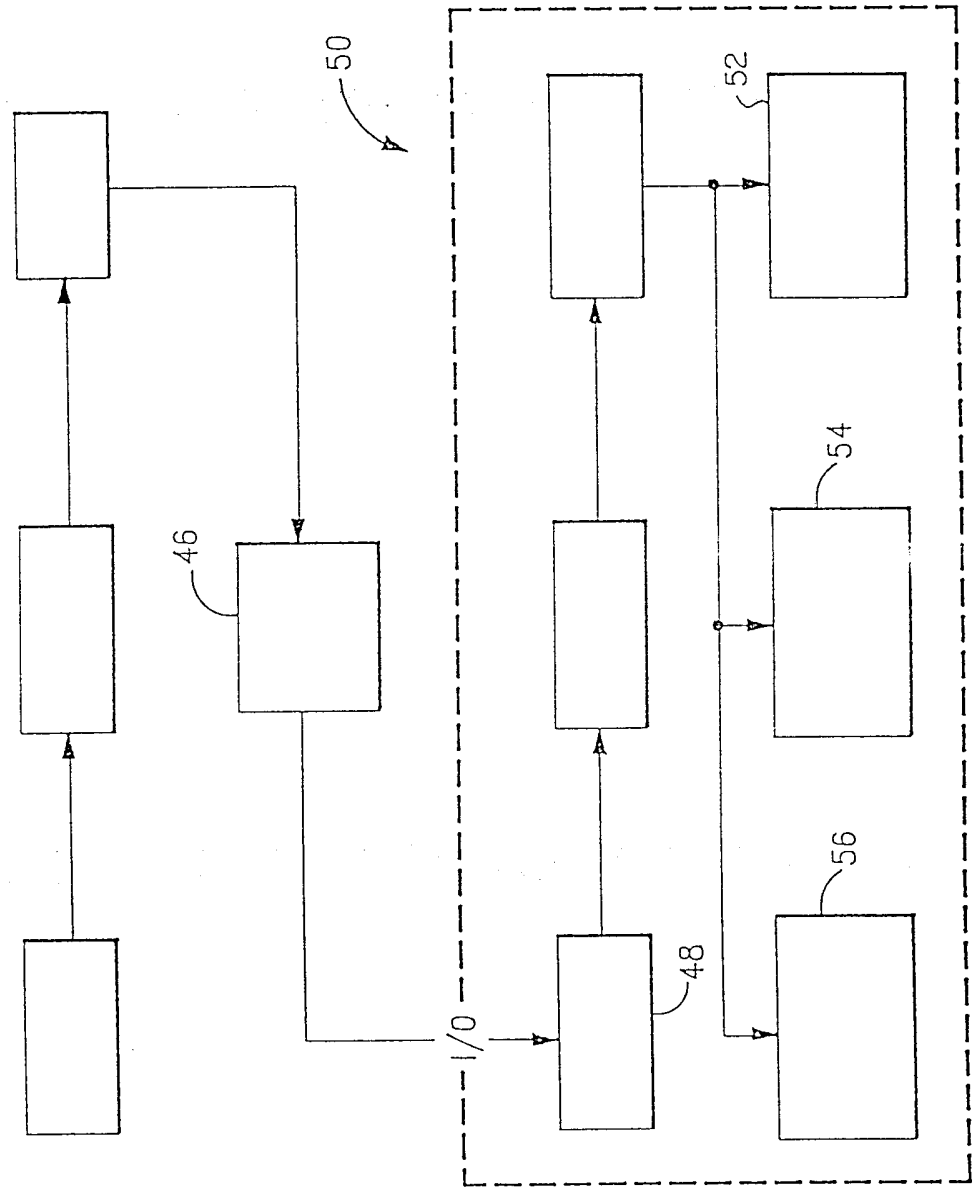


FIG.5

PV 936-91G

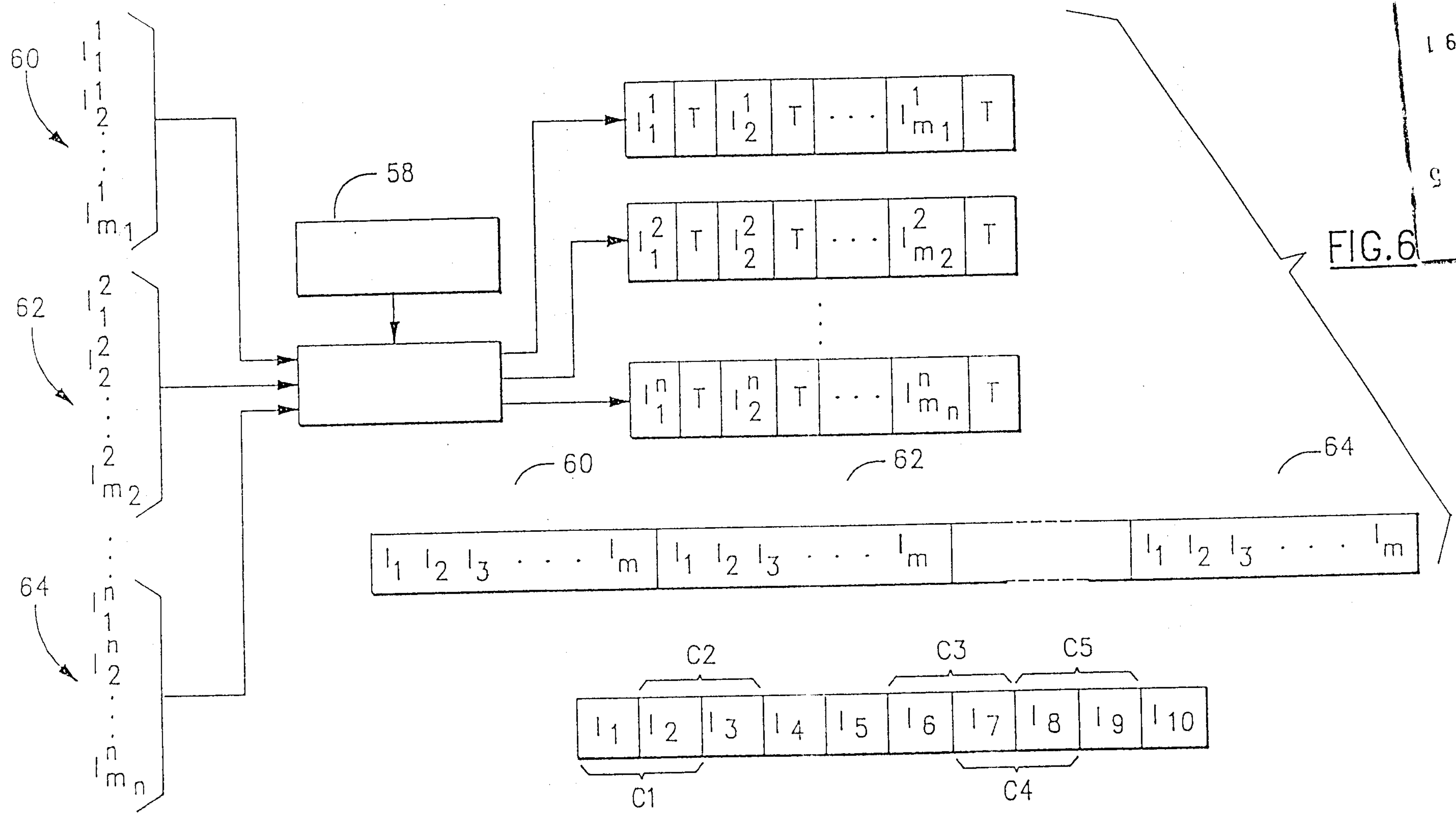


FIG.6

PV 936-91G

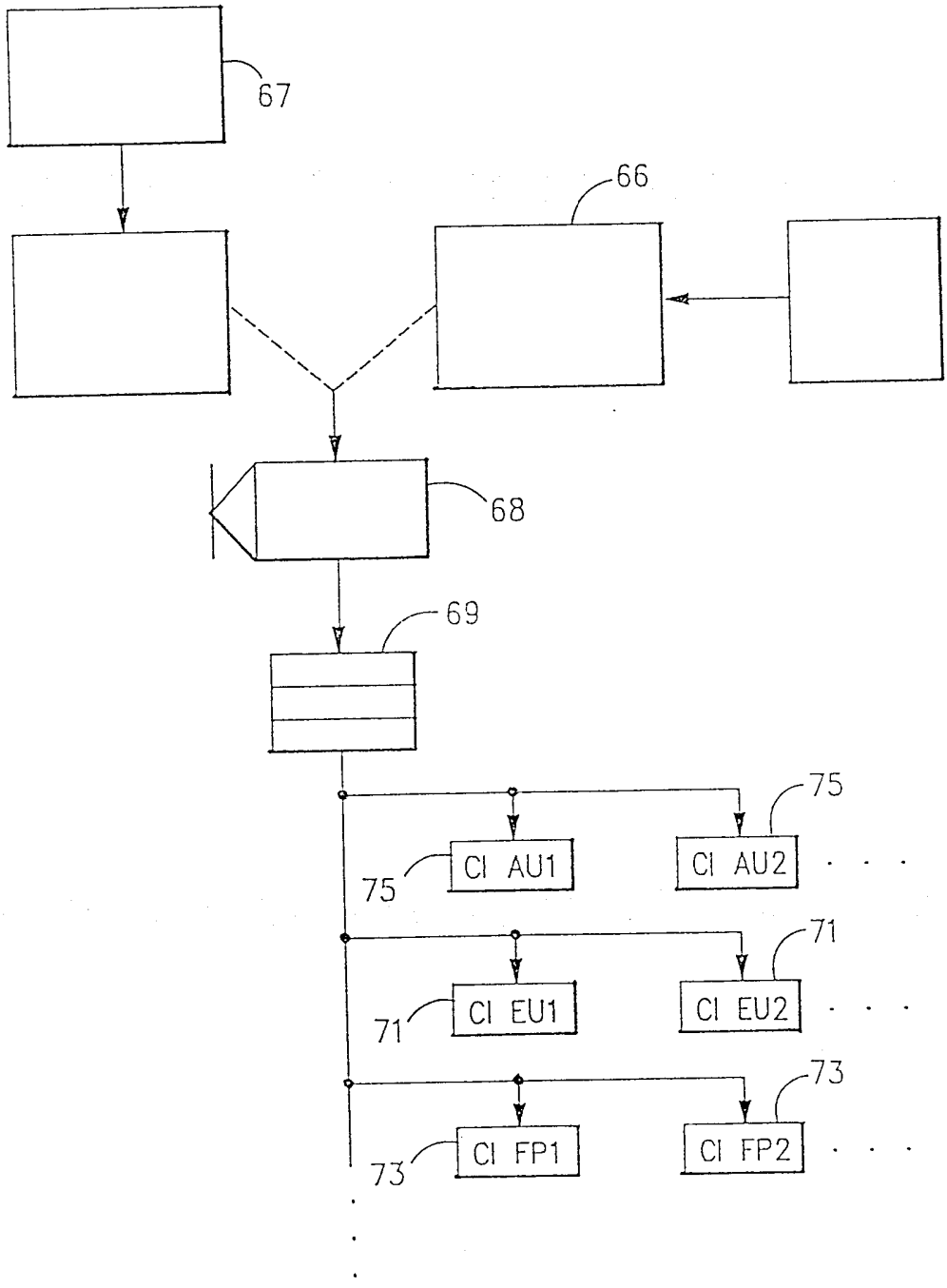


FIG.7

PV 936-91

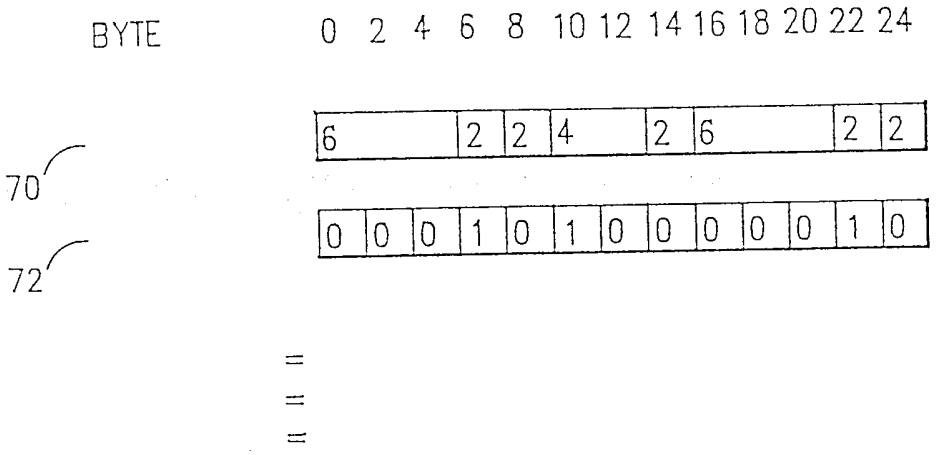


FIG.8

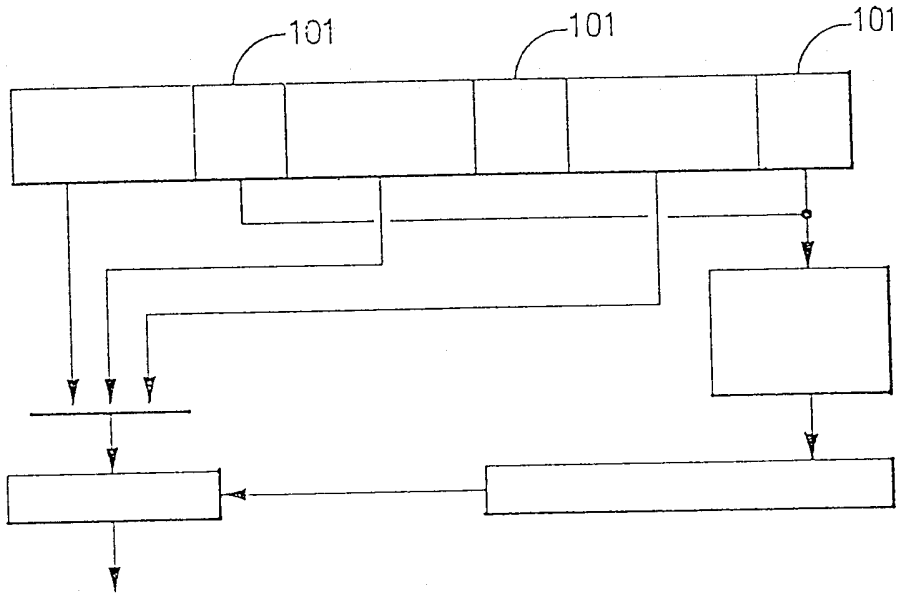


FIG.13

PV 936-91

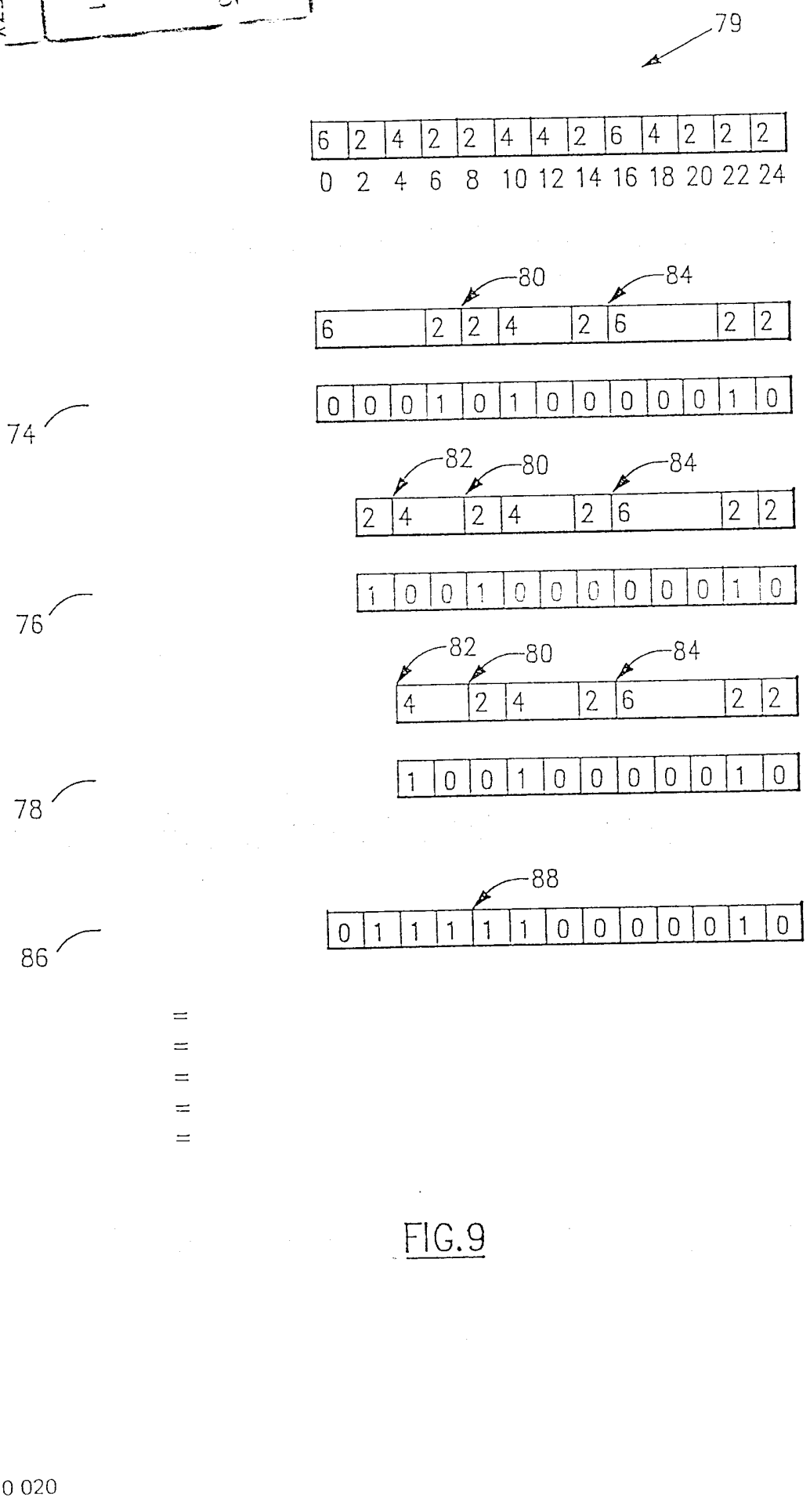


FIG.9

PV 9.36-91G

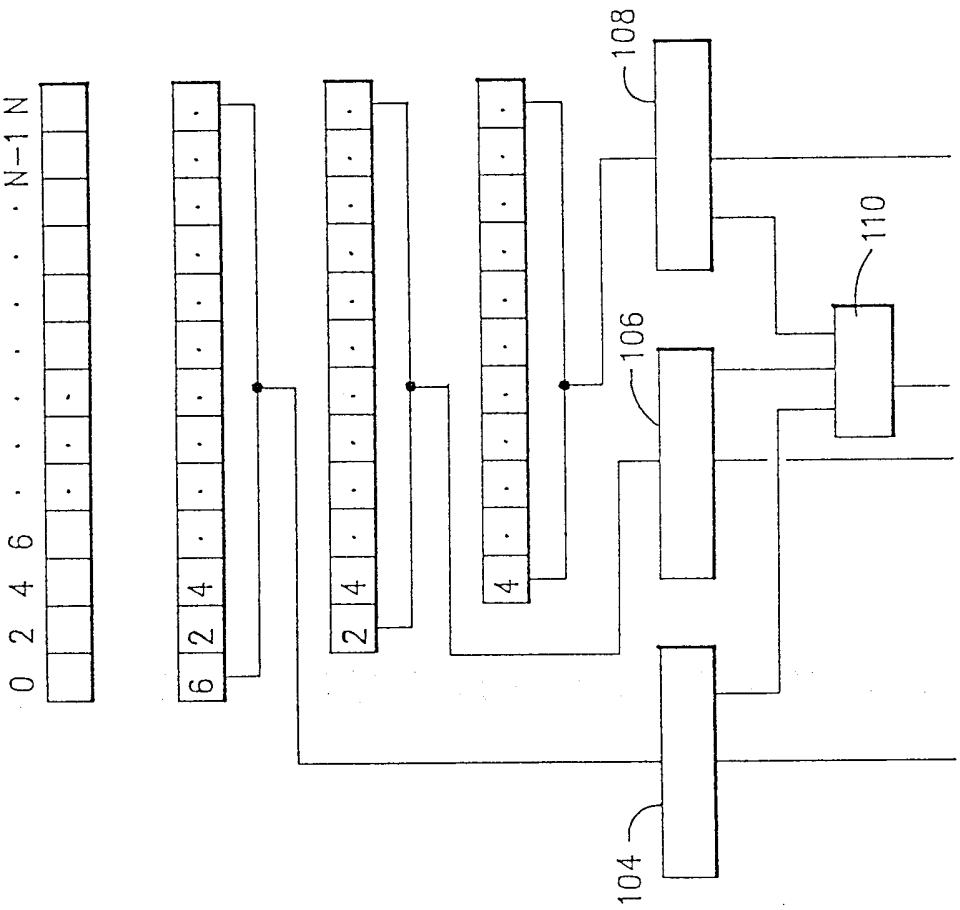


FIG.10

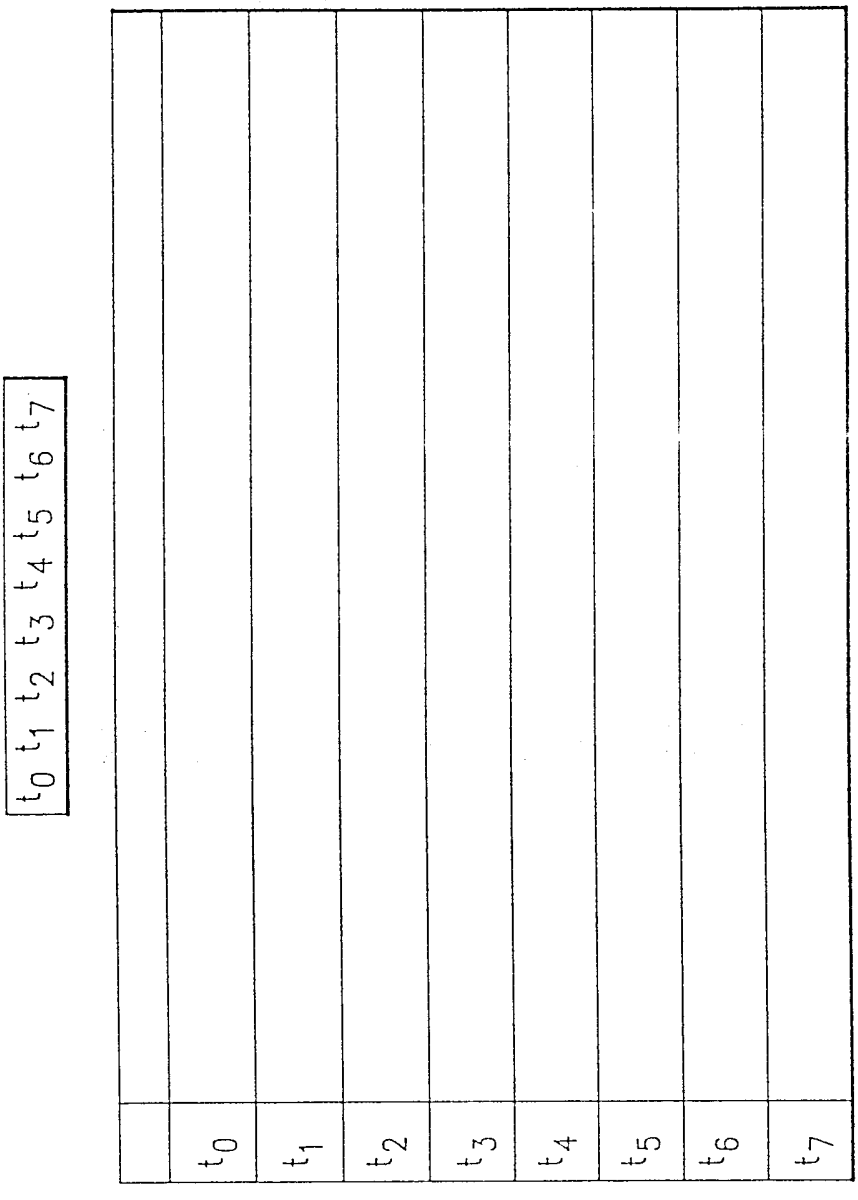
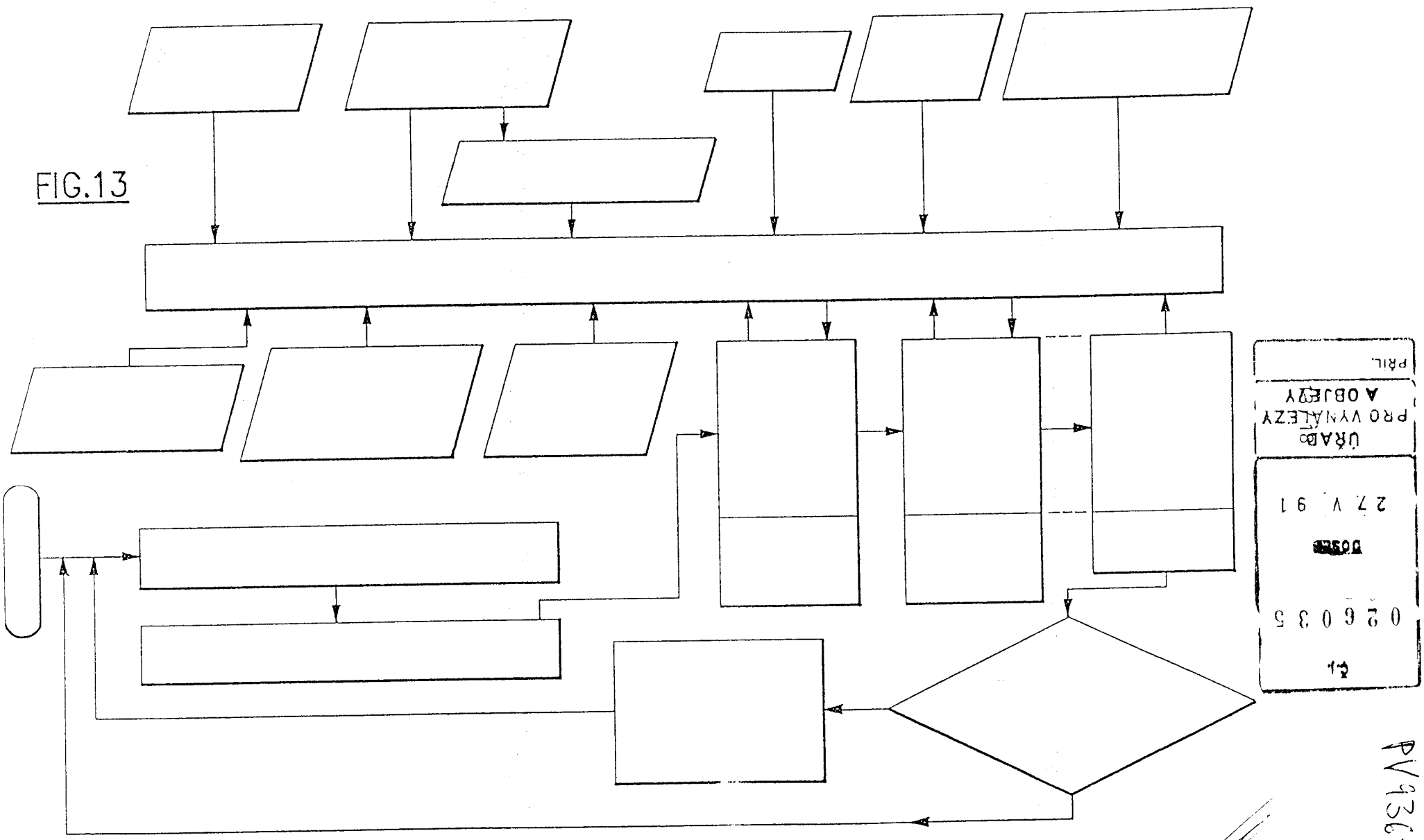
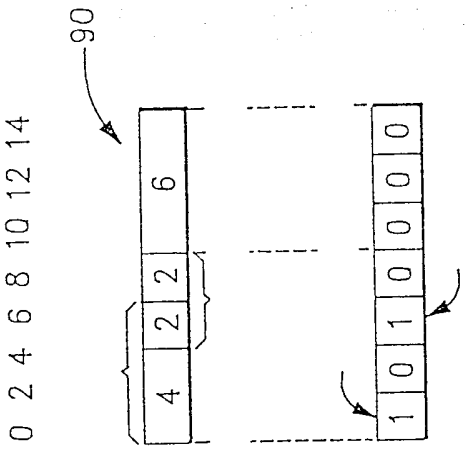


FIG.12

FIG.13



PV 936-91



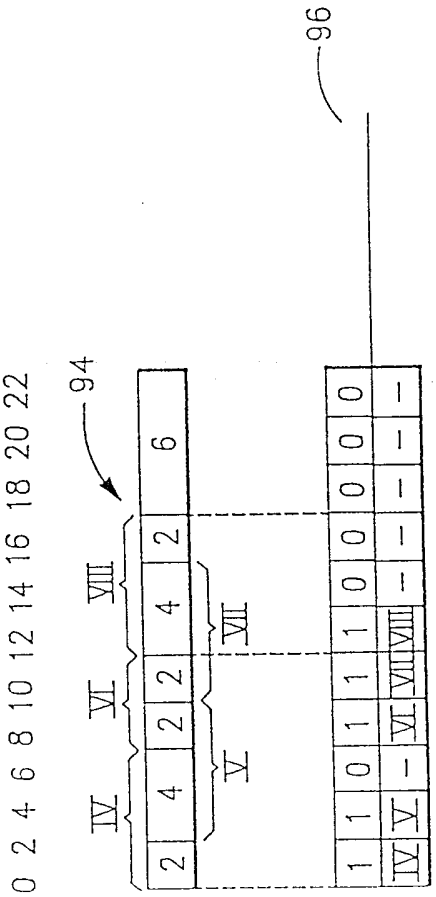


FIG.14

PV 936-916-

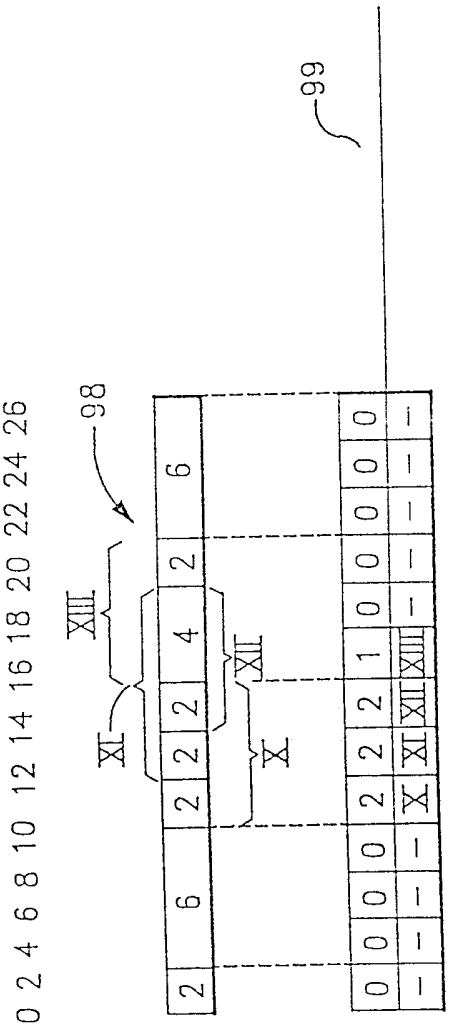


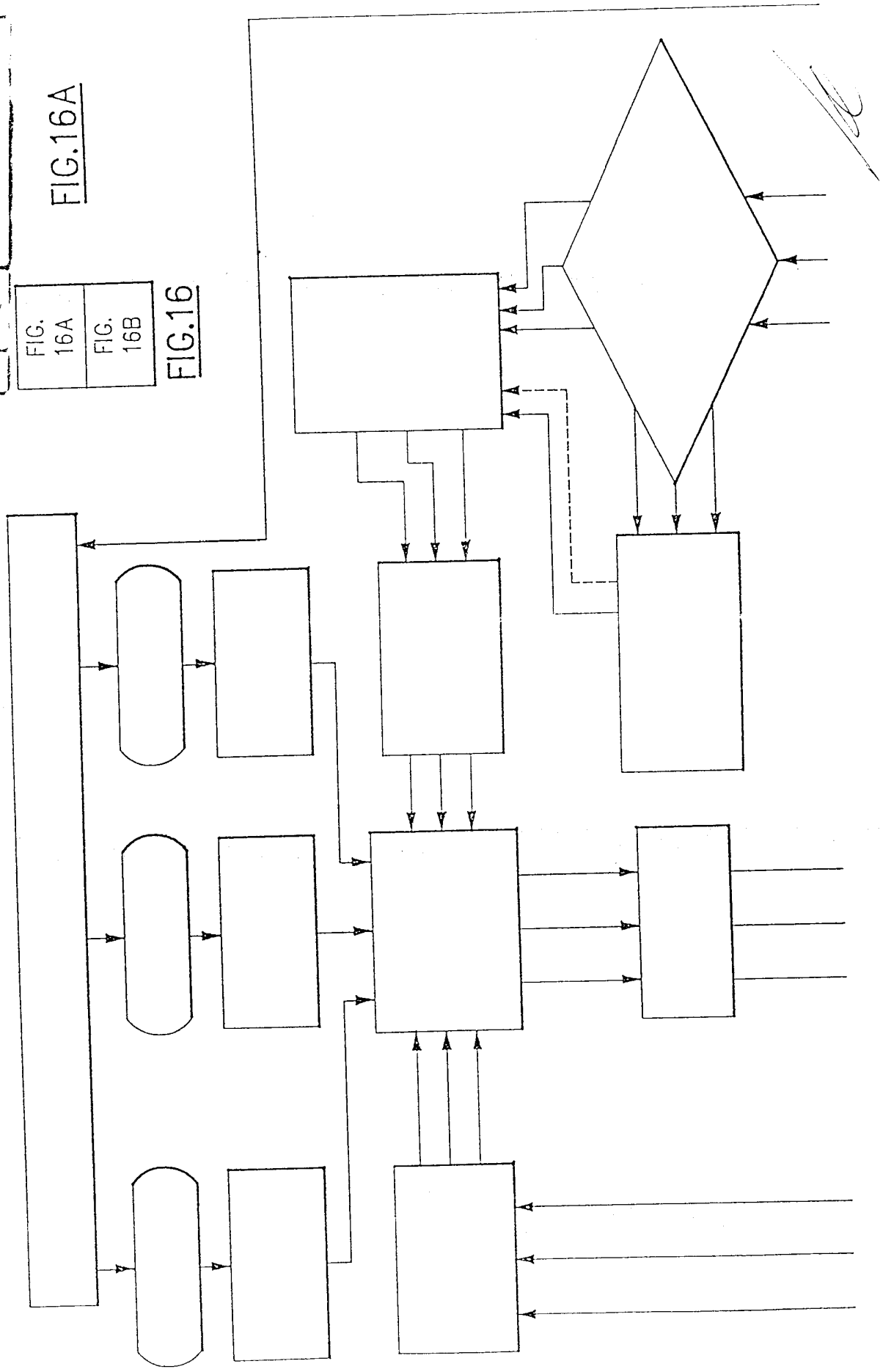
FIG.15

PV 936-91G-

FIG.16A

FIG.16

FIG. 16A	FIG. 16B



PV 936-91C-

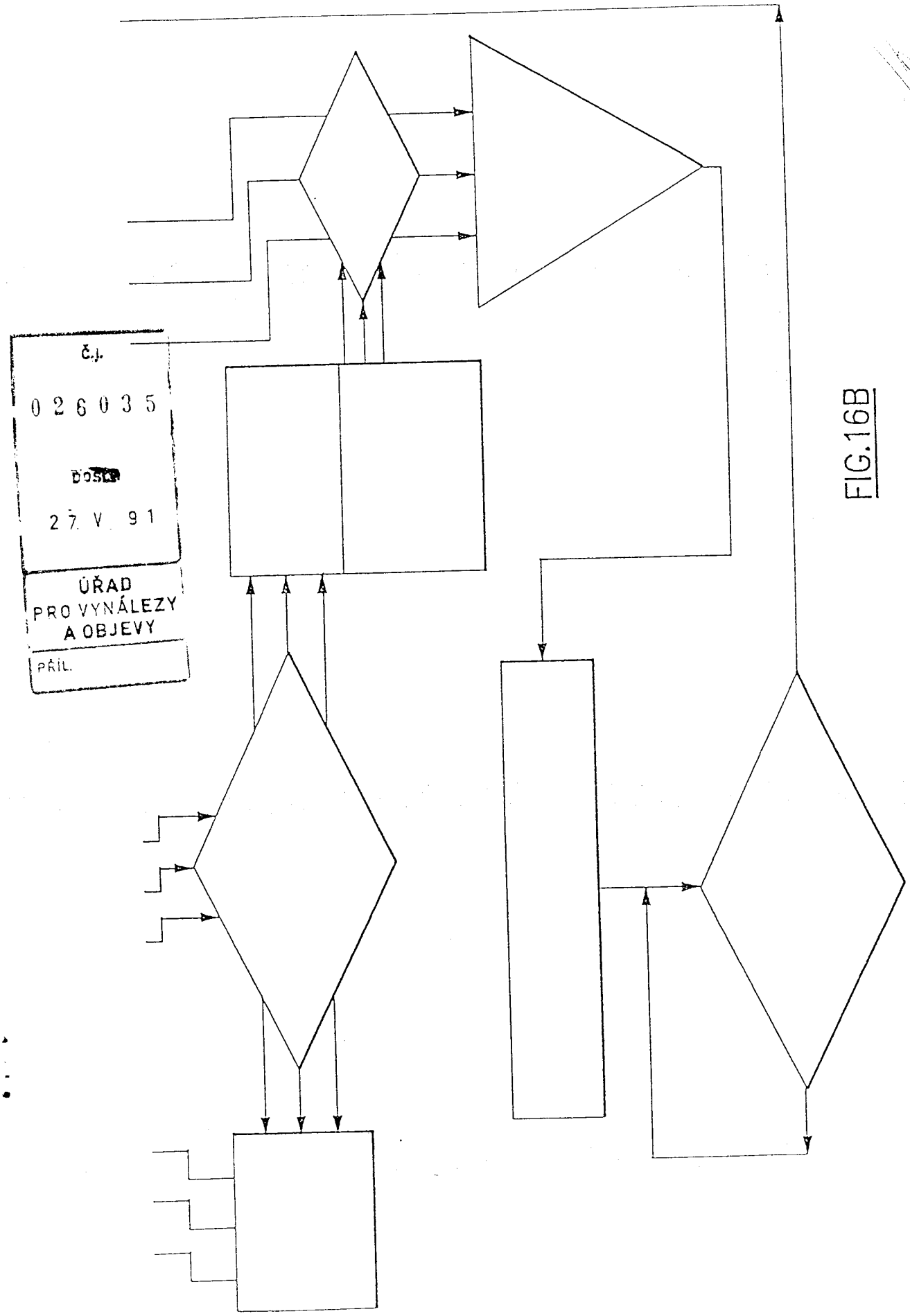


FIG.16B

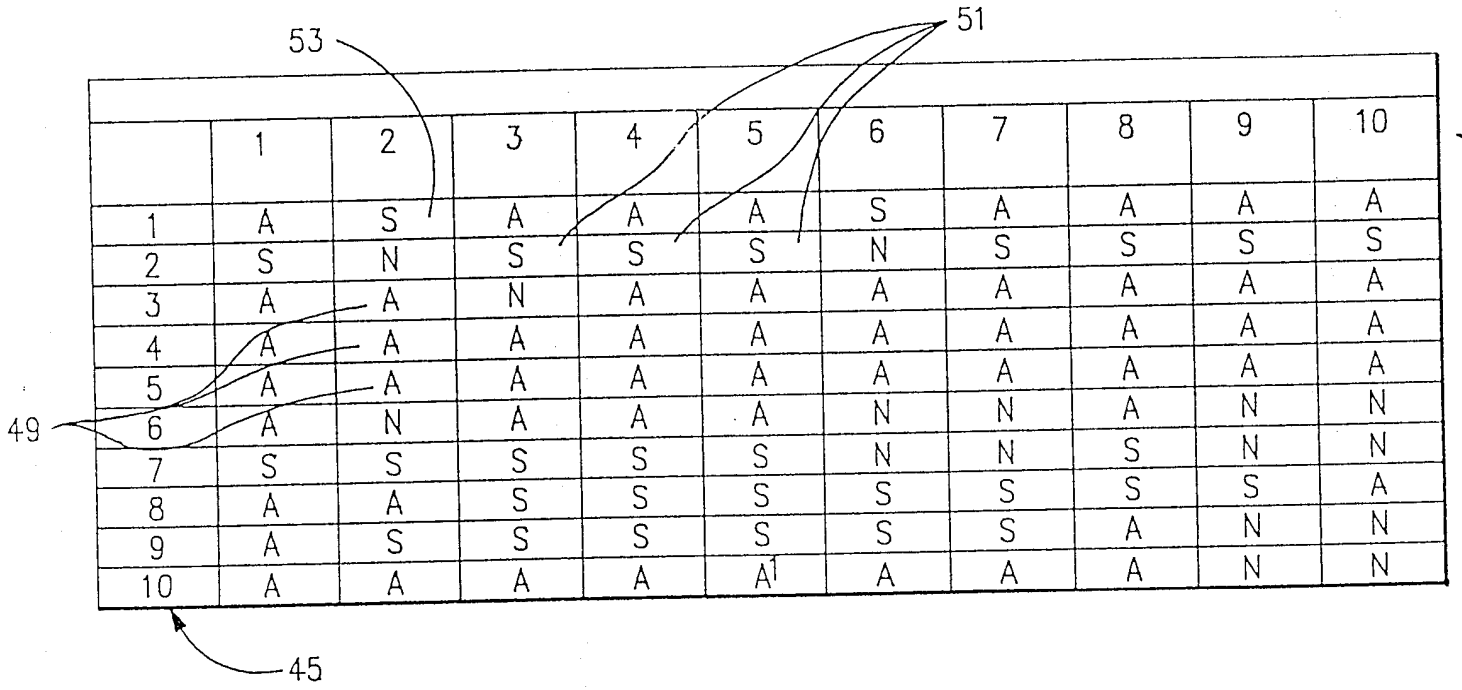


FIG.17

A
S
N

PV 936-916