Architekturentwicklung von Hochleistungsprozessoren

Probevorlesung, Dortmund (20. 3. 91)

- Disposition -

Die folgenden Ausführungen betreffen universelle Digitalrechner des oberen Leistungsbereichs, wie sie bereits heute in Form von Mikroprozessorschaltkreisen zur Verfügung stehen.
Der Zwang zur absoluten Leistungssteigerung ist offensichtlich nach wie vor so massiv, daß die an sich weite Verbreitung eingeführter Architekturen (i..86) nicht davon abschreckt, weitere Innovationen auf dem Gebiet der Architektur (im Ggs. zur Technologie, Taktfrequenz usw.) auszuarbeiten.(Empirischer Sachverhalt, Markt-Beobachtung)
Um unseren Gegenstand näher zu bezeichnen, sei zunächst eine typische Architektur kurz vorgestellt.
Folie 1 i860: anh. Bild kurz erläutern
Ähnliche Prozessoren: Transputer T800, i960CA, Motorola DSP 96002, 88000 u. a.. Auf die allgemeinen Kennzeichen hinweisen: mehrere parallelwirkende Verarbeitungswerke, RISC-ähnliche Befehlsstrukturen, bes. Schaltmittel zur Adressenrechnung, on chip caches.

Die Fragen, mit denen wir uns beschäftigen werden, sind
a) wie sind diese Architekturen entstanden (auf welchen Grundlagen)?
b) welche weiteren Verbesserungen, Ansätze zur Innovation usw. sind absehbar?
Zunächst zur Sinnfälligkeit (warum Befassung mit diesem Gegenstand und nicht einfach Nutzung dessen, was von der Industrie angeboten wird?)
- Wer heute in der Ausbildung steht, kommt in wenigen Jahren in die Praxis; er sollte dort die dann neuesetn Entwicklungen sofort aufgreifen können, also als Profi kompetitiv sein.
Dazu geistiger Vorlauf. "Über den Dingen stehen."
"Nicht alles ist dem Hörenden deutlich zu machen, was dem Ausübenden einleuchtet" (G.).

- Man braucht zumindest Urteilsfähigkeit, um unter den Angeboten auswählen zu können, also mögl. objektive Bewertungskriterien. Deshalb sehe ich ein sinnvolles Ziel darin, auf dem Gebiet der Rechnerarchitektur zu einer objektiven, von Hersteller-Einflüssen unabhängigen technischen Lehre zu kommen (s. Verbrennungsmotoren, Aerodynamik usw.)
- Um die künftige Entwicklung abschätzen zu können, braucht man möglichst viele Idden-Ansätze, auch unkonventionelle. Man kann so abschätzen, welche potentiellen Verbesserungen überhaupt noch zu erreichen sind und ob sich weitere, aufwendigere Bemühungen überhaupt lohnen (angesichts Inkompatibilität usw.)

Zunächst betrachten wir einige Grundlagen, auf denen moderne Hochleistungs-Architekturen aufbauen.

Folie 2 -- Liste der 4 Ansätze; kurz erläutern

Die ersten beiden sind traditionell, die letzteren ergeben sich gleichsam von selbst, wenn man nach weiteren Möglichkeiten zur Leistungssteigerung sucht.
(Hinweisen, wo die Prinzipien beim i860-Beispiel verwirklicht sind.)

Sowohl zur Beurteilung existierender Architekturen als auch als Teil eines Gesamt-Rahmens der angesprochenen technischen Lehre sollen nun 4 Prinzipien vorgestellt werden.

Folie 3 --- Liste der 4 Prinzipien, nur nennen.

Die Überlegungen entsprechen einem "back to the roots approach" (Schopenhauer: Zum Philosophieren sind die zwei ersten Erfordernisse diese: erstlich, daß man den Mut habe, keine Frage auf dem Herzen zu behalten, und zweitens, daß man alles, was sich von selbst versteht, sich zum deutlichen Bewußtsein bringe, um es als Problem aufzufassen.)

Beginnen wir damit, die Selbstverständlichkeit als Problem aufzufassen, daß ein Digitalrechner, ~~daß jeder Di~~ aus Verarbeitungs- und Steuerschaltungen, Speichern, Datenpfaden usw. besteht, mit einem Wort als eine Sammlung von Ressourcen angesehen werden kann.

Folie 4 -- Computer as a collection of....

kurz erläutern.

Die Allgemeingültigkeit dieses Schemas ist daran ersichtlich, daß es die Extremfälle einschließt: v.Neumann, Datenfluß.

Ressourcenausstattung an sich bestimmt in erster Linie die Verarbeitungsleistung. Formel Leist. = Ress-Zahl/Masch. zyklus

Dann Speicher + Datenpfade

Erst an 3. Stelle die Befehsliste.

Dazu müssen wir etwas ins Detail gehen: Register-Transfer-Niveau: ist allen Architekturen gemeinsam, auch den ausgefallensten; jeder Rechner besteht letzlich aus binären Schaltmitteln und stellt sich aus dieser Sicht so dar:

Folie 5: Ressourcenvektor

Die systematische Vorgehensweise wird offensichtlich:

1. Ressourcen-Vorrat (Inventory): Ress.-Algebra an-deuten
2. konkr. Prozessor durch optimierende Auswahl+ Datenweg-Gestaltung usw.
3. Befehlsstruktur kann systematisch behandelt werden: Ziel ist, daß die Ress. möglichst nur nützliche Arbeit leisten (jeder Masch. zyklus nützlich z. Lösung der Anw. aufgabe, keine "Hilfszyklen").

Offensichtliche Lösung: Zuführung aller Steuer- und Datenangaben parallel.

Folie 6 Speisung des Ress. vektors

näherungsweise bei VLIW- und Datenflußarchitek.

Das ist aber nicht immer durchführbar.

Den üblichen Kompromiß erläutern, Hinw. auf RISC/CISC.

Künftig möglicherweise Ress.-Algebra als primäres Compiler-Ziel.

Objektorientierung

Notwendig zur Beherrschung großer Softwarekomplexe.
Folie 4 -- Grundschema des obj. orient. Datenzugriffs

Hardware-Unterstützung derzeit aus der Mode. Durchbruch durch massiven Hardware-Einsatz: das Schema direkt vergegenständlichen
Folie 8 Speicherschema des objektor. Zugriffs.

Wir sehen hier etwas ungewöhnliches, nämlich eine Ansammlung von Speichern verschiedener Aufrufbreite und Kapazität. Abweichung vom üblichen linearen Adressenraum.

Solche Anordnungen beruhen auf dem nächsten Prinzip:

Kontrollierte Kardinalität

An sich nichts neues: alle Ressourcen sind grundsätzlich endlich in ihrer Anzahl (Addierwerke, Speicherzellen usw.). Diese Endlichkeit ist vor dem Nutzer sorgfältig zu verstecken (Compiler, Sprachen, Laufzeitsysteme, Betriebssysteme usw.) Auf der Ebene der Maschinenbefehle (bzw. des Ress. vektors) gibt es grundsätzlich zwei Ansätze:
1. knapp, gerade hinreichend, Nutzung bleibt der Softw. überlassen (z. B. Register-Allocation)
2. unausschöpfbar groß (übl. Speicheradressenraum, virtueller Speicher) .

kurz diskutieren
Endlichkeit der Ress. der Software zu Optim.zwecken zur Verfügung stellen, dazu empirische Befunde nutzen
(z. B. Parameterzahl in Programmen)

Vergegenständlichte Abstraktionen

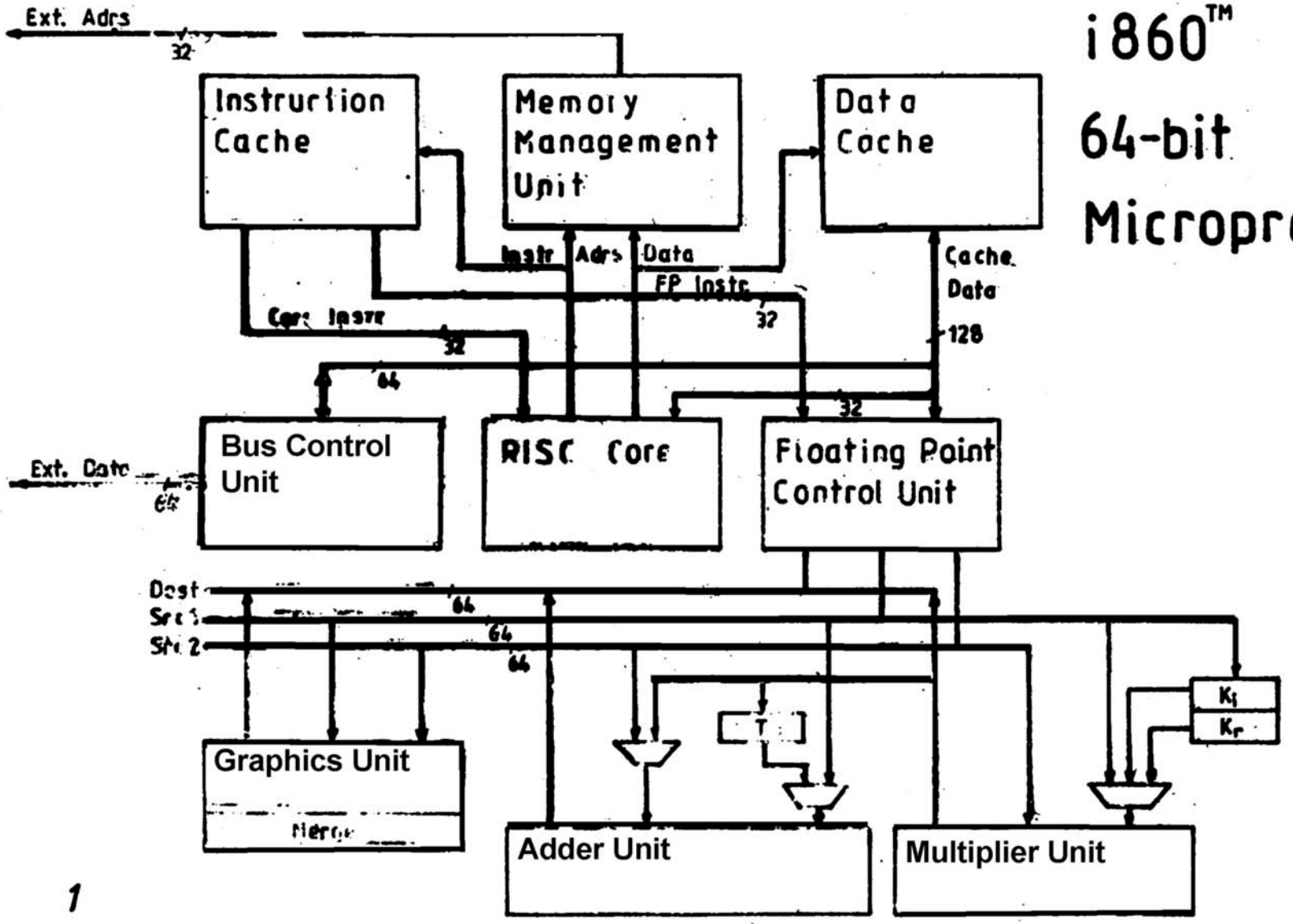
Grundlagen-Wissenschaften und andere Erfahrungsquellen systematisch absuchen nach Prinzipien, die sich eignen, in Hardware realisiert zu werden.

Erweiterung der Erfahrungsbasis über den instruction level hinaus. Statistische Analysen allein können dazu führen, daß heutige u. vergangene Programmiergepflogenheiten (Fortran, Cobol) in künftige Architektur-Generationen weitergeschleppt werden und daß potentielle Innovations-Gelegenheiten gar nicht erst erkannt werden.
Einfaches Beispiel 1: Iterator
Folie 9 3-stuf. Iterator, kurz erläutern
Einfaches Beispiel 2: Elementar-Mathematik
Folie 10: 1 Bsp. f. Datenfluß-Graphen
Resultat: 4 univ. Op-Werke können sinnvoll beschäftigt werden (bei instruction level anlysis nur 2!).
Folie 11/12 Übersicht über künft. Prozessorstruktur

Folie 12 Aufbau eines Operationswerkes (Ressourcensammlung)

---falls Zeit ist, ggf. die Verb. strukt. (f. Anfragen Folie vorrätig halten).

i860™ 64-bit Microprocessor



Ansätze zur Architekturgestaltung

1. Optimierung der Maschinenbefehle

2. Nutzung des innewohnenden Parallelismus

---

3. Ganzheitliche Betrachtungsweise

4. Entwurfsgrundsätze von Spezialmaschinen

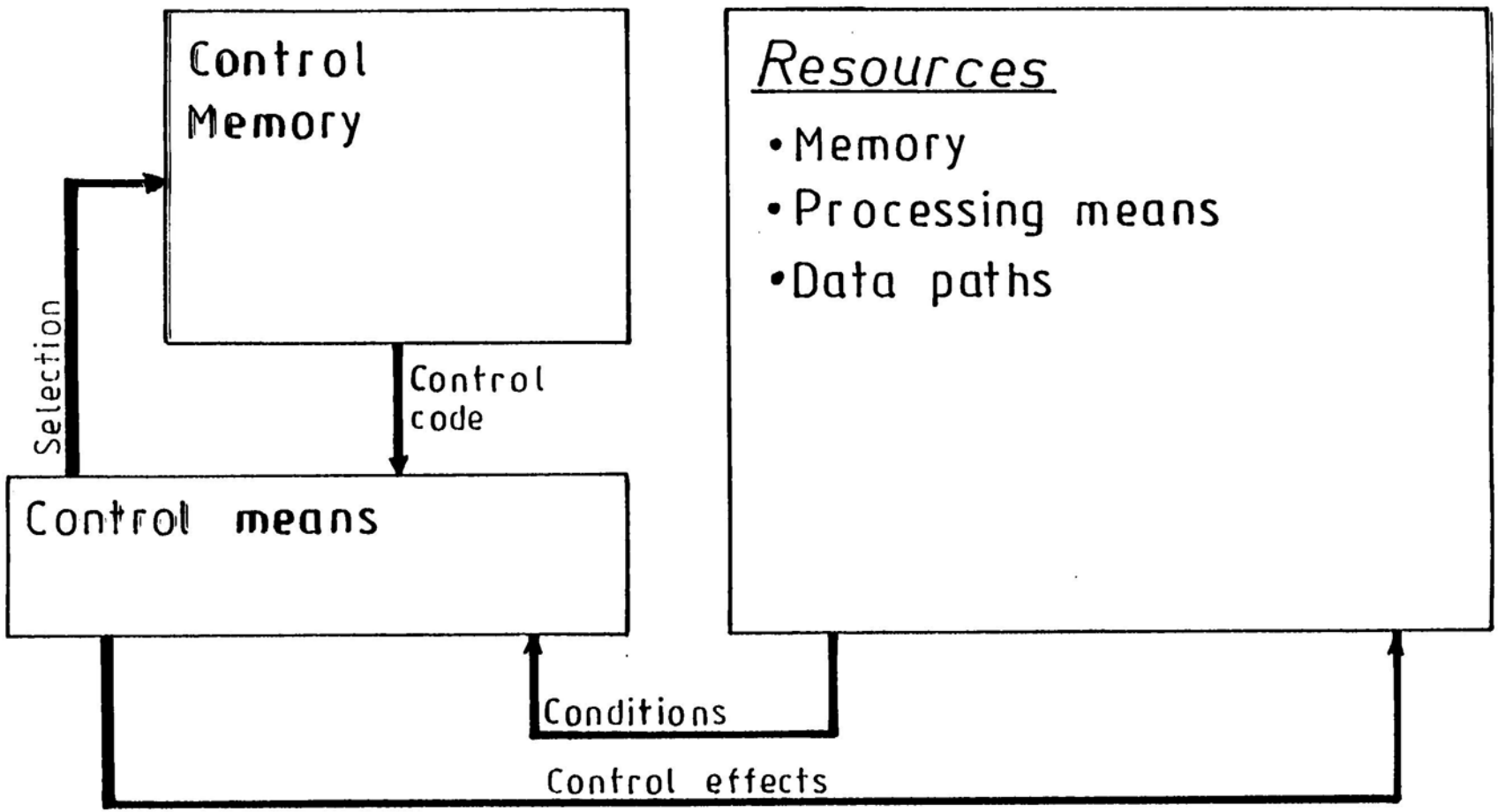
Prinzipien

1. Ressourcen-Paradigma

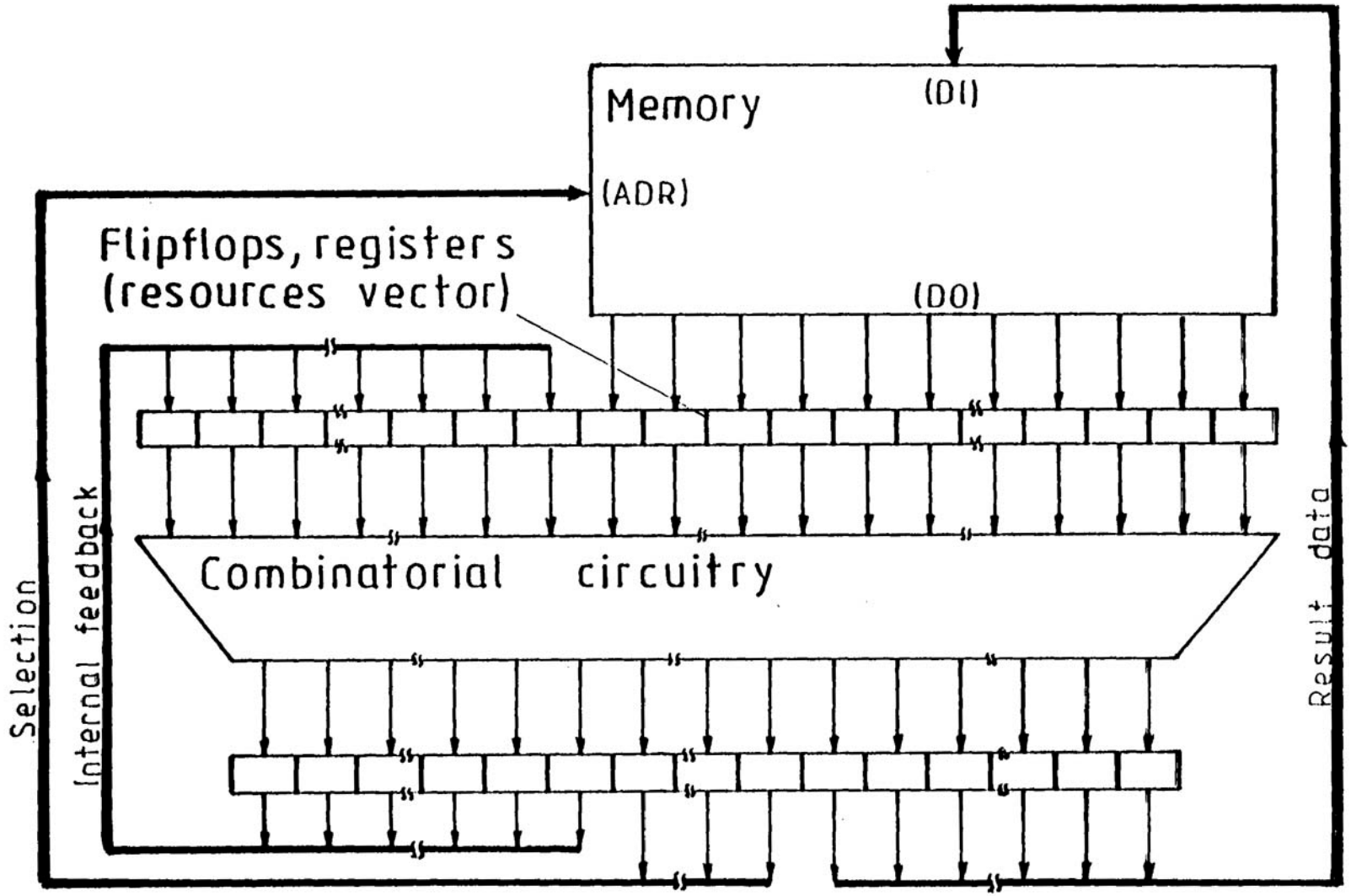
2. Objektorientierung

3. Kontrollierte Kardinalität

4. Vergegenständlichte Abstraktionen

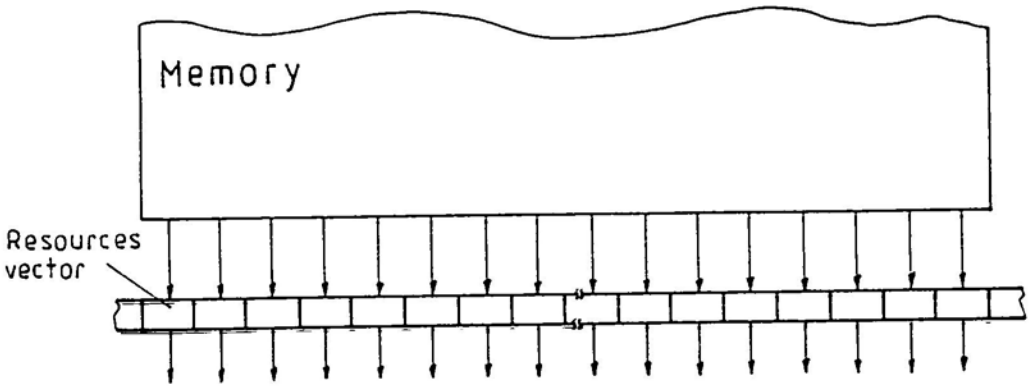


Computer structure as a collection of resources.

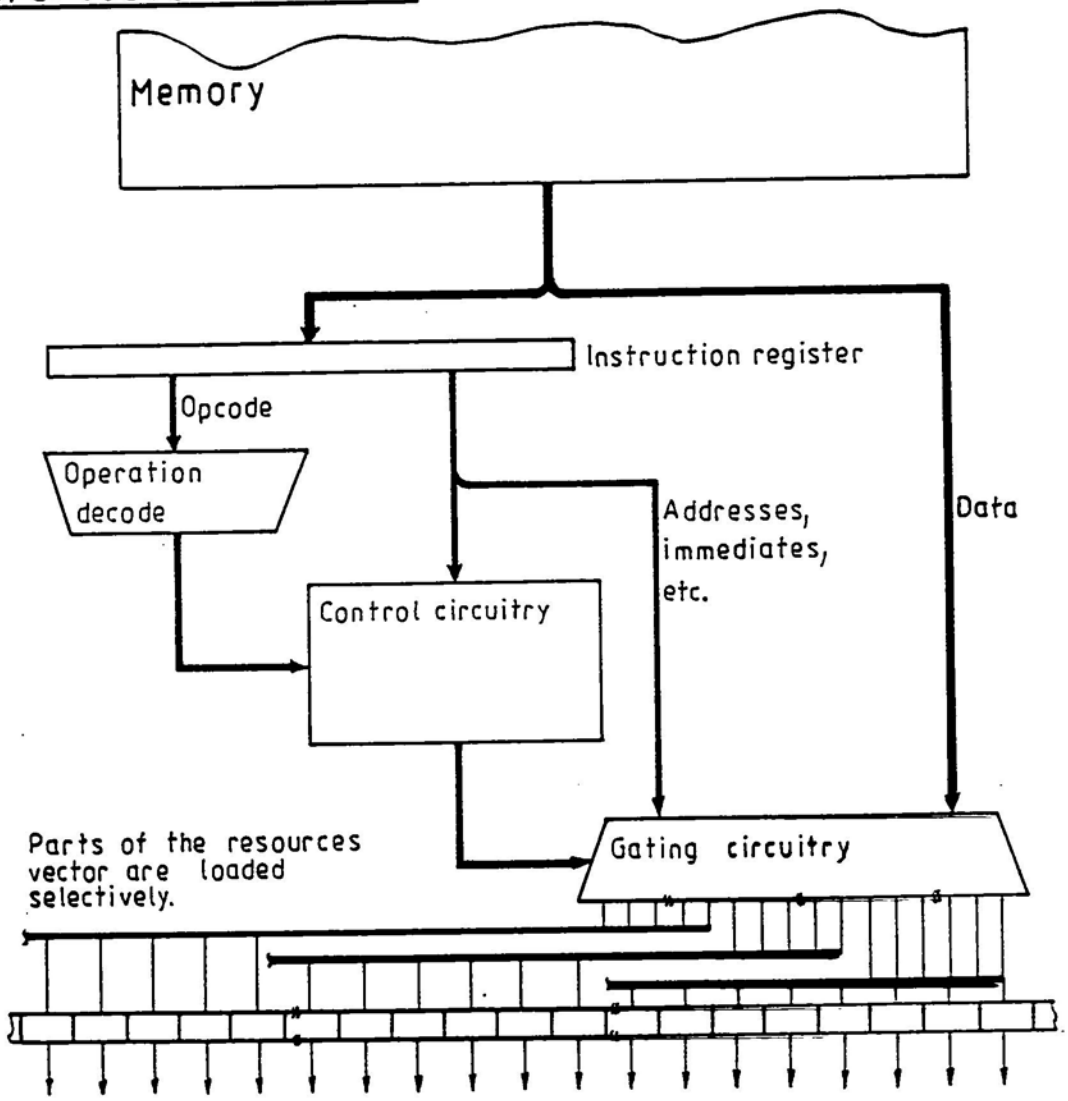


In-detail view: the resources vector.

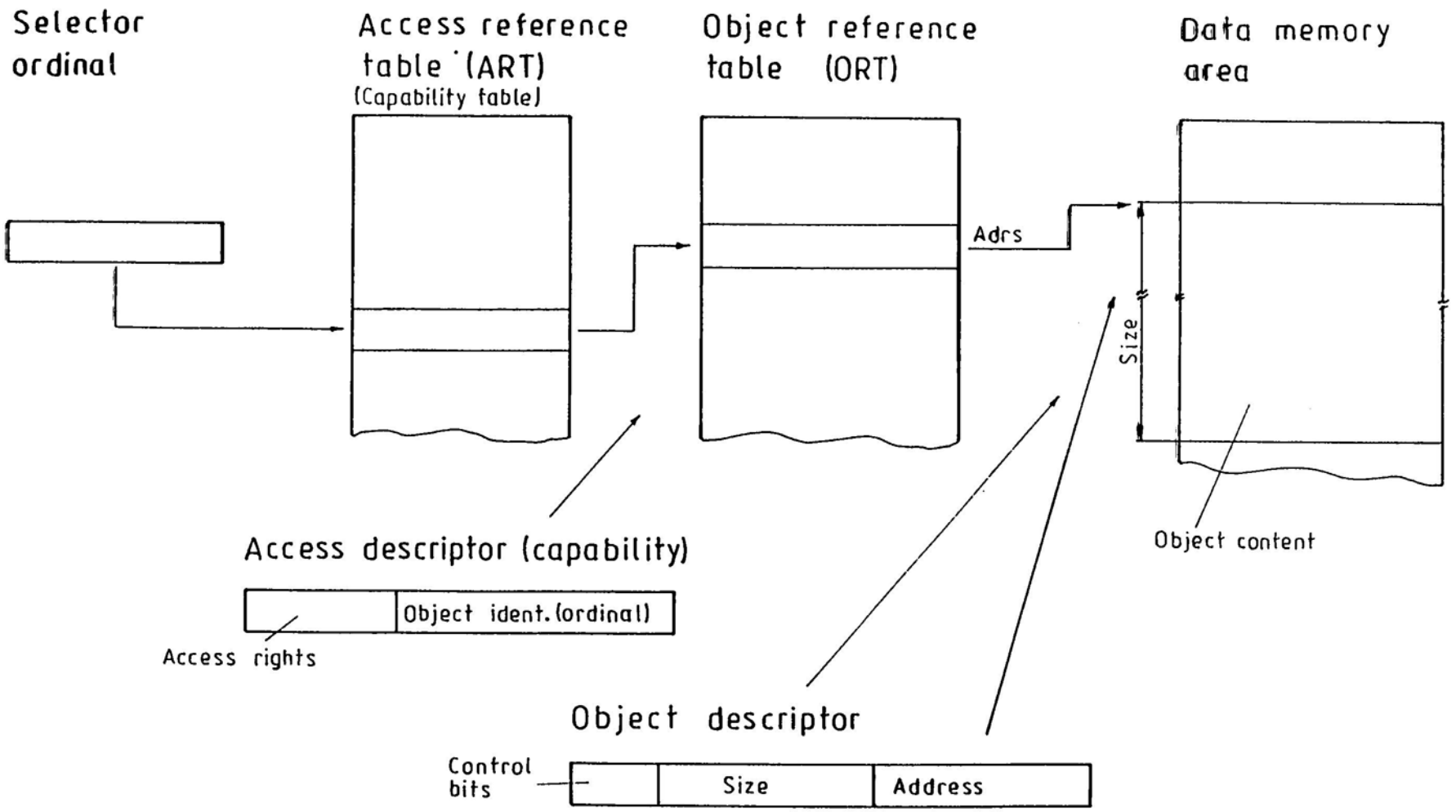
a) Immediate assignment



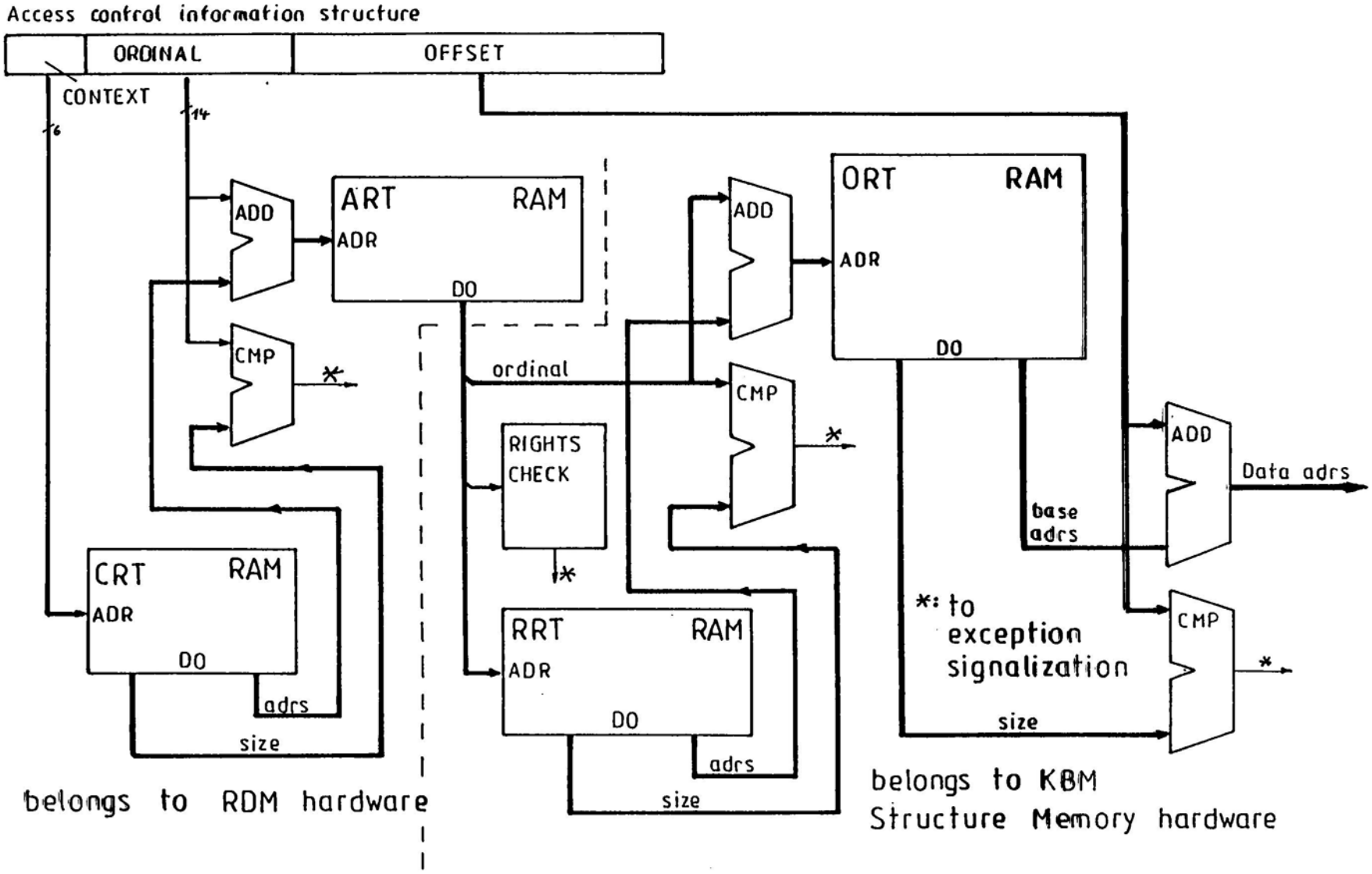
b) Encoded selection



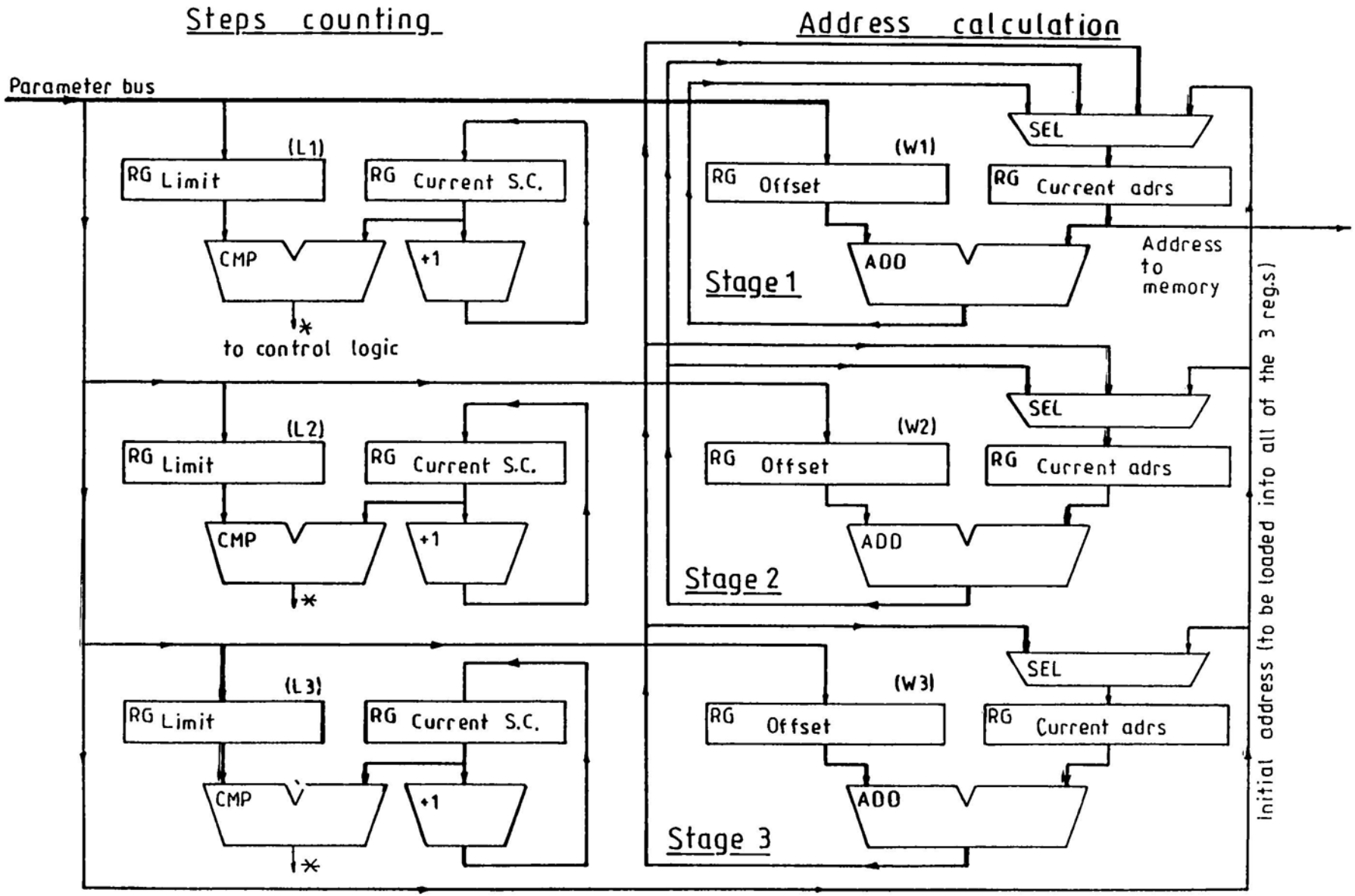
Alternatives: feeding the resources vector out of the memory.



Object-oriented access scheme.



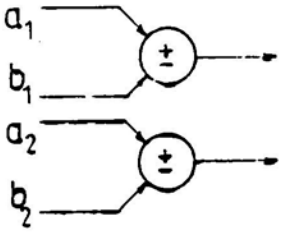
8 Hardware incarnation of object oriented data access.



Iterator hardware for three nested DO-loops. (Stage 1 corresponds to innermost loop.)

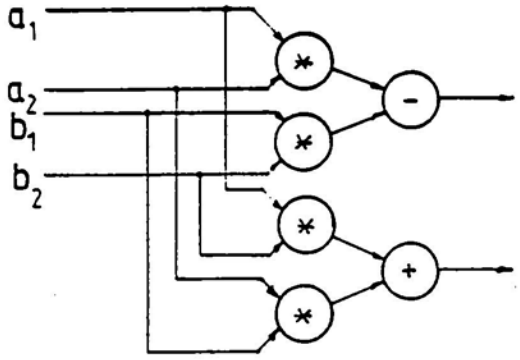
Addition/subtraction

$$(a_1, b_1) \pm (a_2, b_2) = (a_1 \pm a_2, b_1 \pm b_2)$$



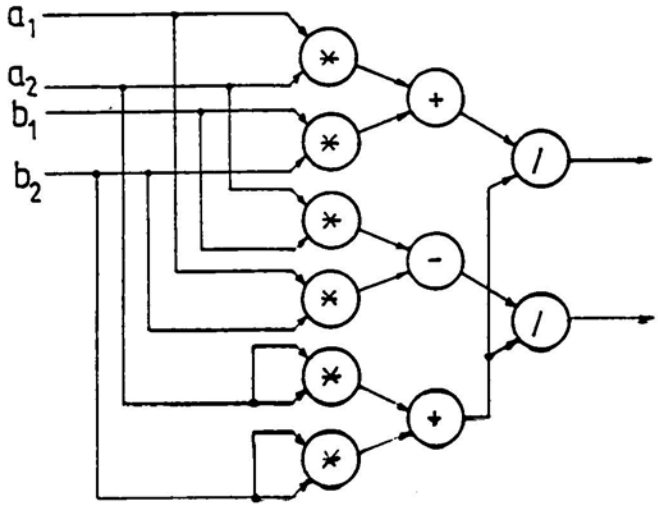
Multiplication

$$(a_1, b_1) * (a_2, b_2) = (a_1 a_2 - b_1 b_2 \,,\, a_1 b_2 + a_2 b_1)$$



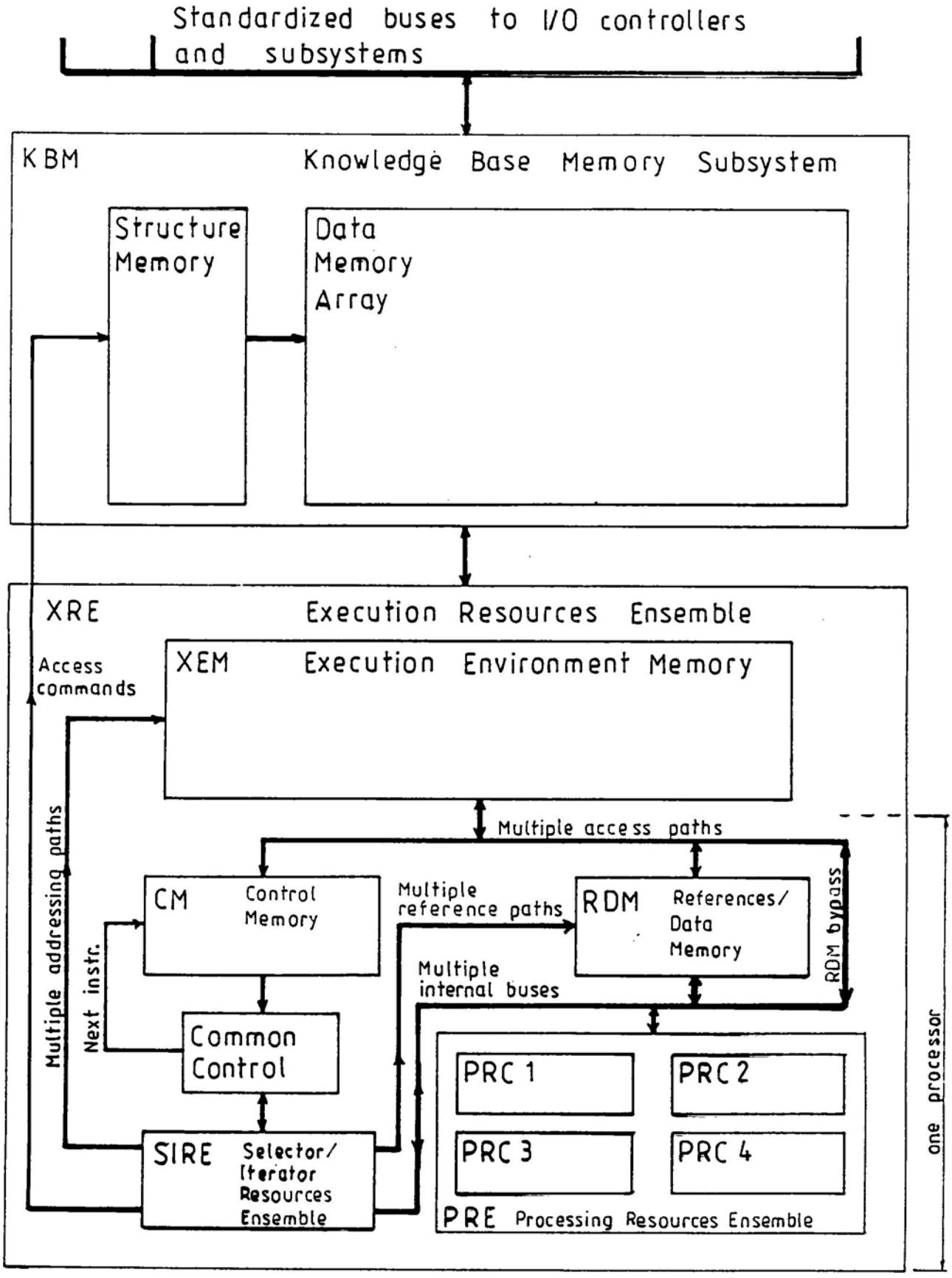
Division

$$(a_1, b_1) / (a_2, b_2) = \left[\frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} \,,\, \frac{a_2 b_1 - a_1 b_2}{a_2^2 + b_2^2} \right]$$

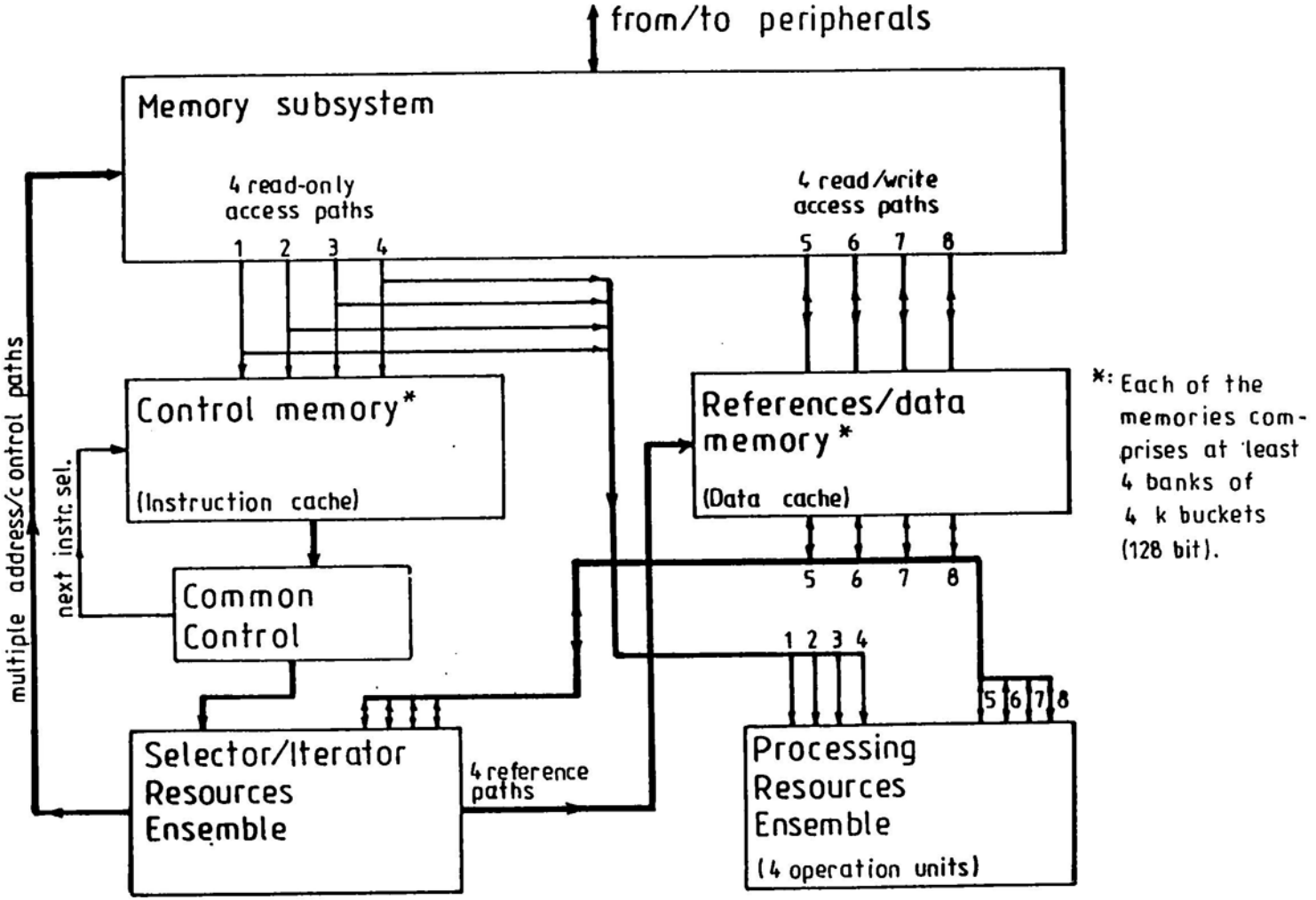


10

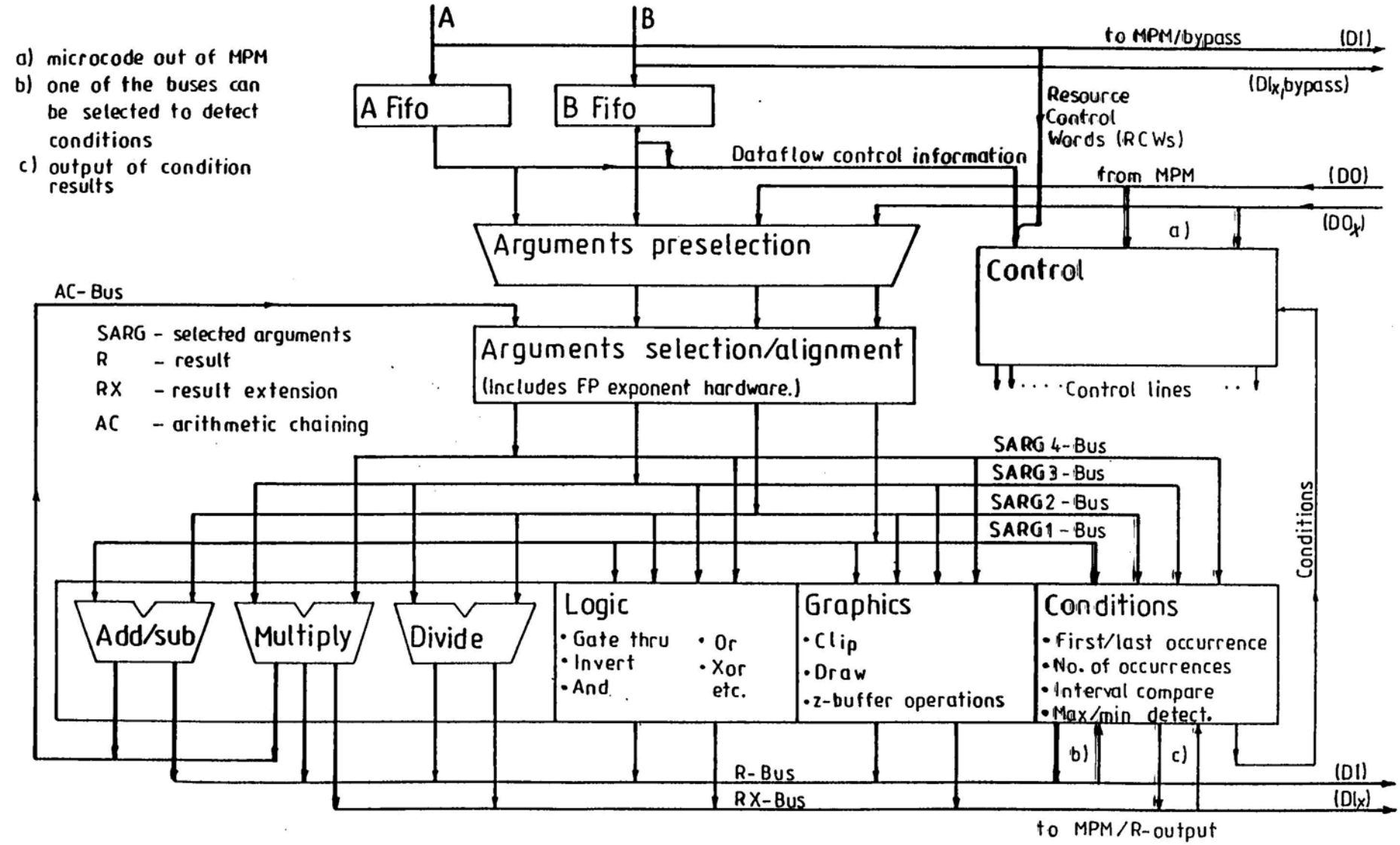
Examples of elementary complex number arithmetic operations.



System overview.



Processor structure.



13 Internal structure of a processing kernel.