CS2022 Microprogrammed Control

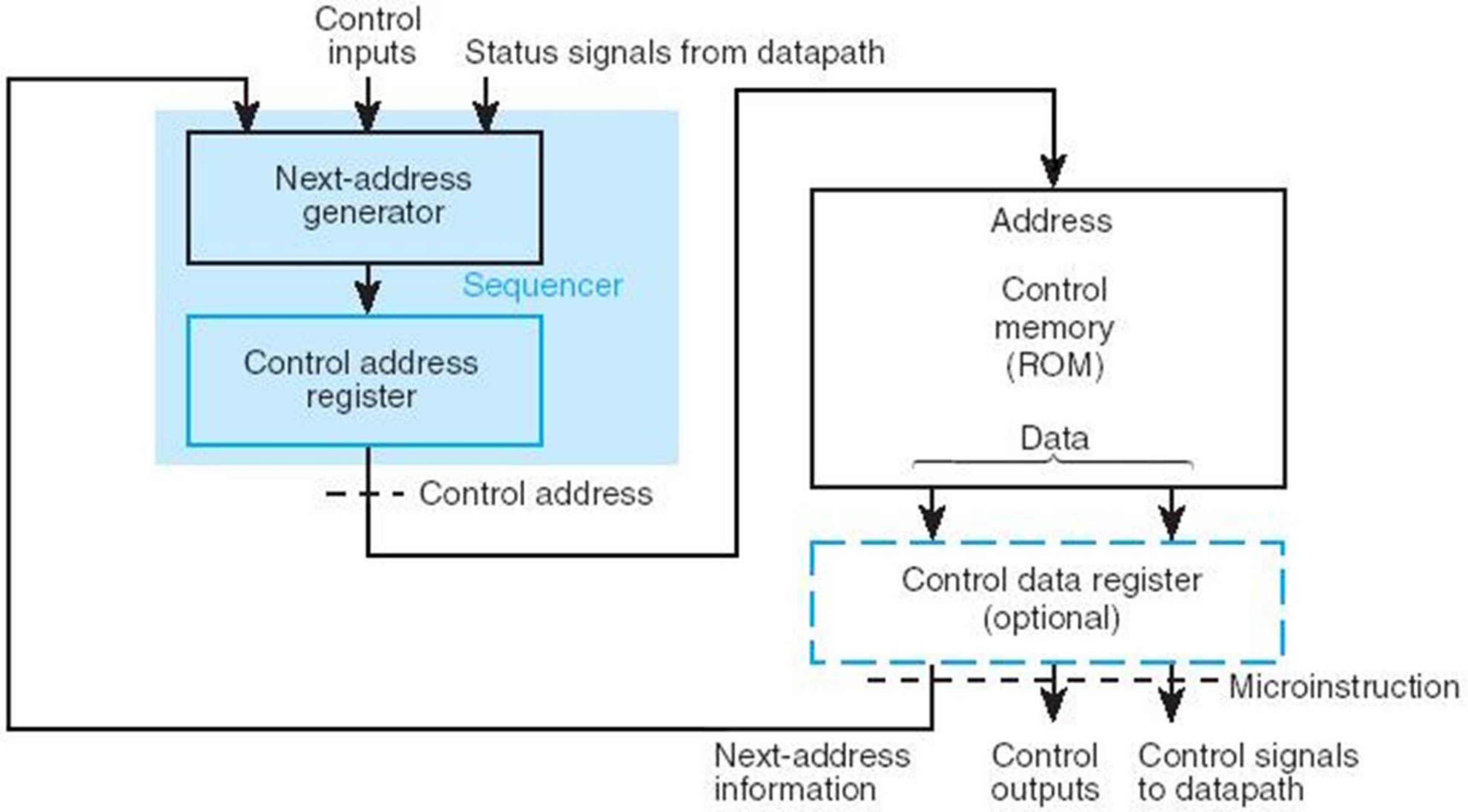
- Hardwired control units have the advantage of great compactness and low propagation delay on account of the shorter loop path.
- However as the number of states and control signals increases they become increasingly costly to:
  - Design
  - Debug
  - Upgrade
- A more flexible approach is used.

CS2022 The Flexible Approach

- We store the control words, together with next-state information , in a control memory which is usually:
  - ROM
  - EPROM
- Then use the control inputs and status signals to select the appropriate address of the next state.
  - See figure on the next slide.

CS2022

Microprogrammed Control Unit Organization

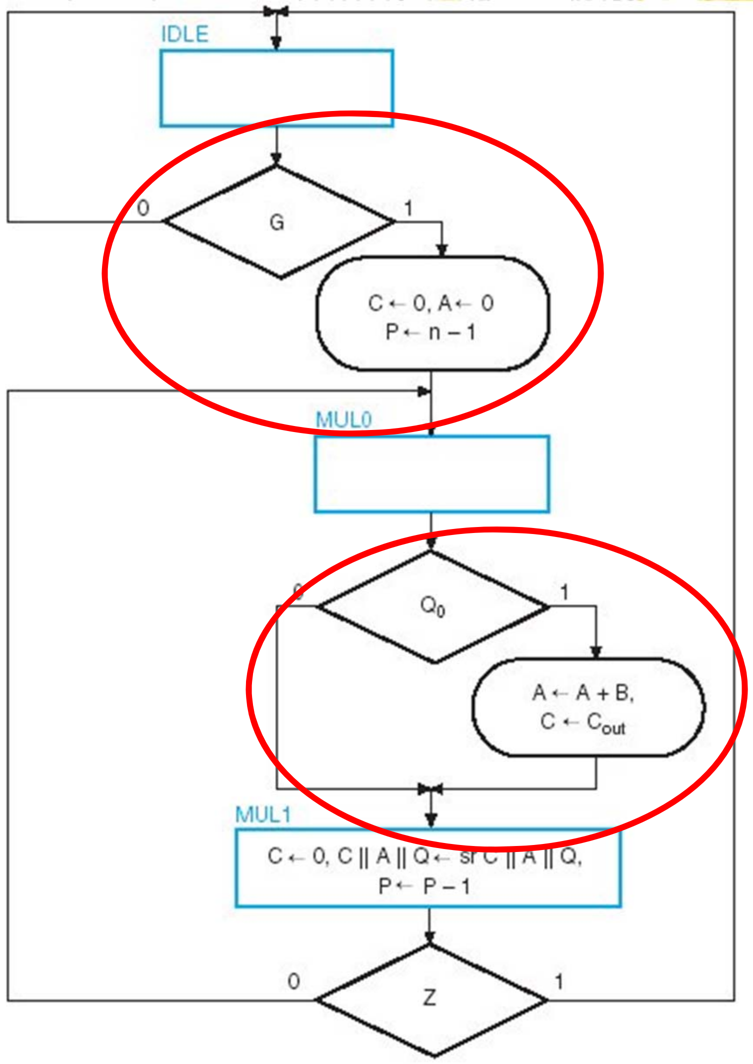


CS2022 Control Input

- One difference which emerges is that since control words are now stored, they cannot be made to depend dynamically on the on the value of the control input

STATE	Control Input	RT	Control Word
IDLE •	G=0	none	CW_1
IDEL •	G=1	$C, A \leftarrow 0$	CW_2
MUL0 •	Q_0=0	none	CW_3
MUL0 •	Q_0=1	$A \leftarrow A+B,\ C \leftarrow C_{out}$	CW_4
more		more	more

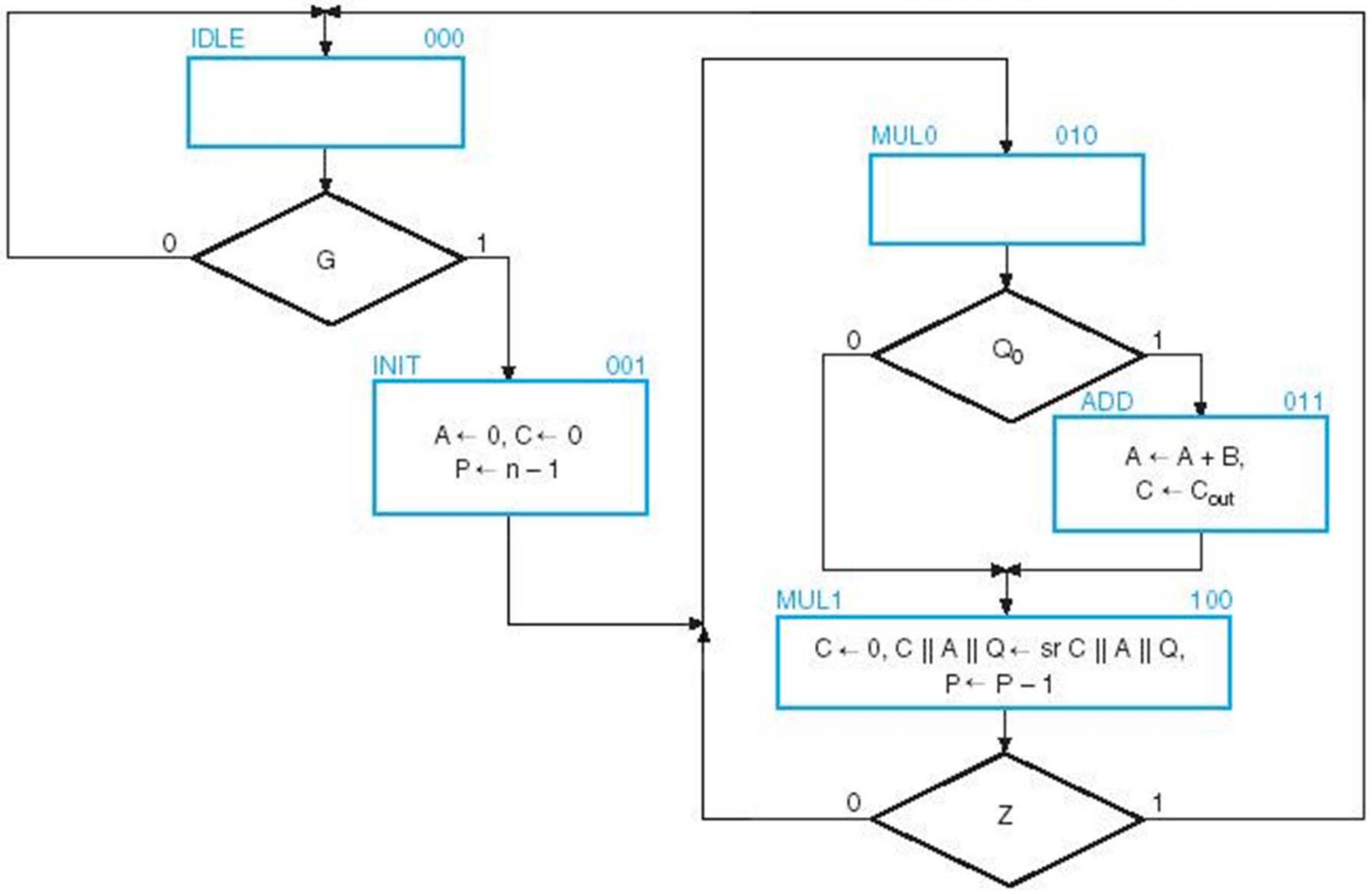
CS2022 Binary Multiplier ASM



- Hence we must introduce additional states to supply these alternative control words.
- See next slide

Microprogrammed Control Unit Binary Multiplier

CS2022



CS2022 Control Address Register (CAR)

- This gives five states so that the control memory must store five control words
  - We will need a 3-bit address register
    - Control Address Register (CAR)

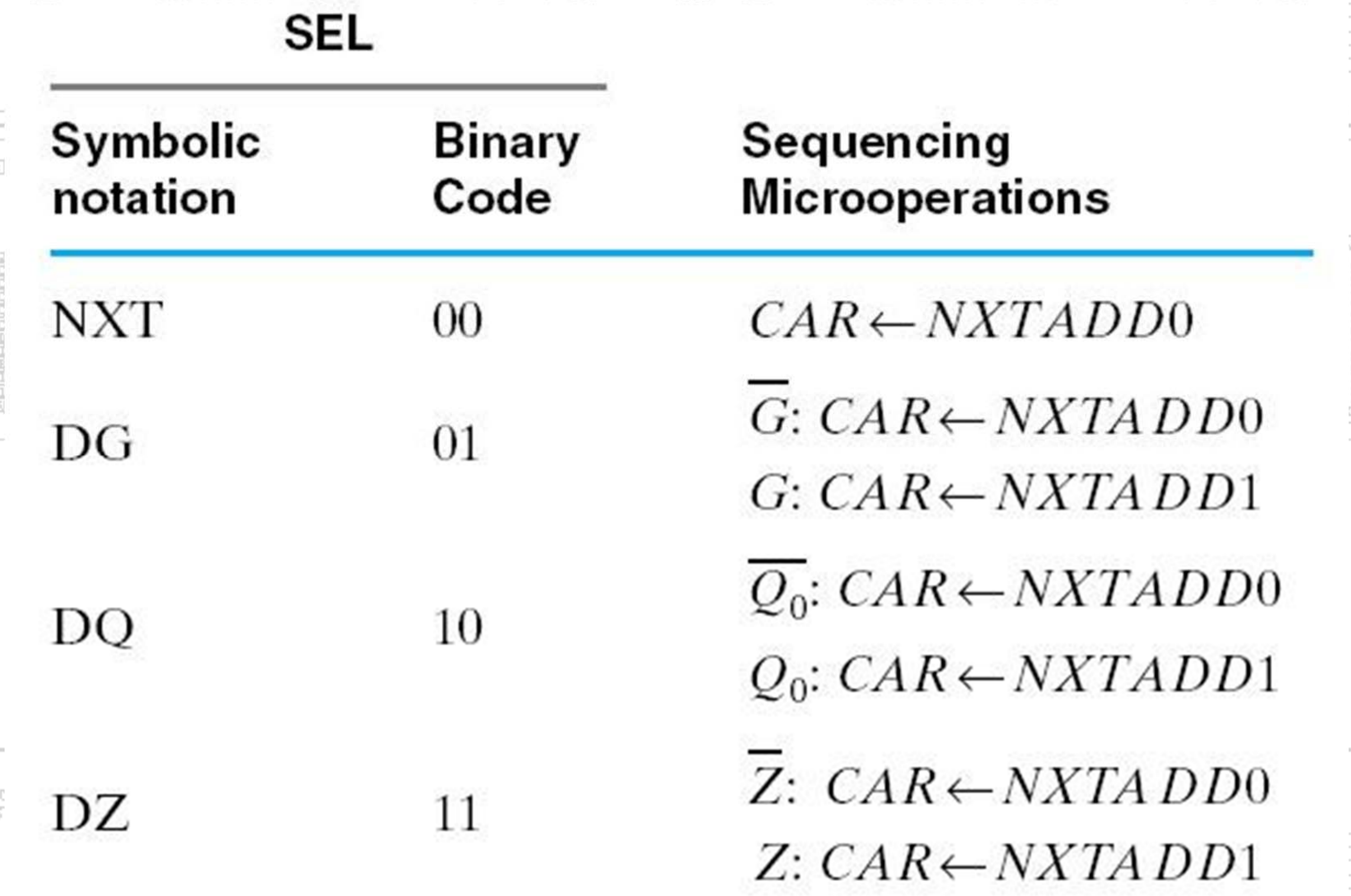
CS2022 SEL bits

- The sequence must be able to respond to two status bits:
  - Z
  - Q_0
- One control input:
  - G
- Hence two SEL bits must be included in the control word.

CS2022 ## Next-Address Fields

- Finally we add two next-address fields:
  - NXTADD0
  - NXTADD1
- This design makes no assumption about the sequence control word accesses.
- The functional design of the sequence is as follows on the next slide:

CS2022 SEL Field Definition

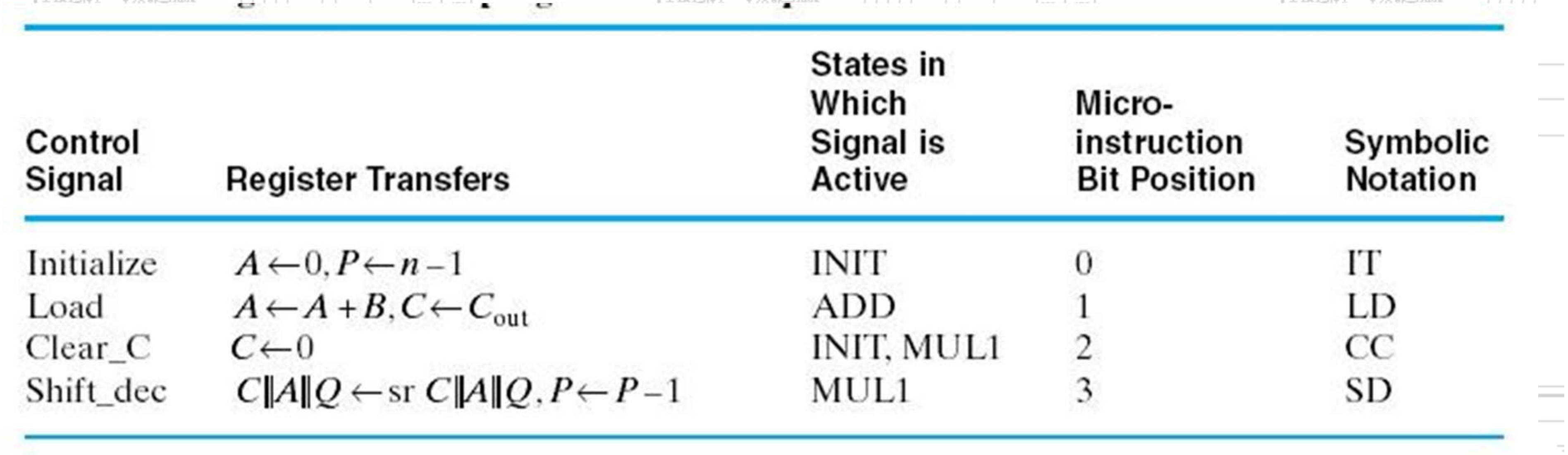


SEL

Symbolic notation	Binary Code	Sequencing Microoperations
NXT	00	$CAR \leftarrow NXTADD0$
DG	01	$\overline{G}: CAR \leftarrow NXTADD0$ $G: CAR \leftarrow NXTADD1$
DQ	10	$\overline{Q_0}: CAR \leftarrow NXTADD0$ $Q_0: CAR \leftarrow NXTADD1$
DZ	11	$\overline{Z}: CAR \leftarrow NXTADD0$ $Z: CAR \leftarrow NXTADD1$

Control Signals for Multiplier control

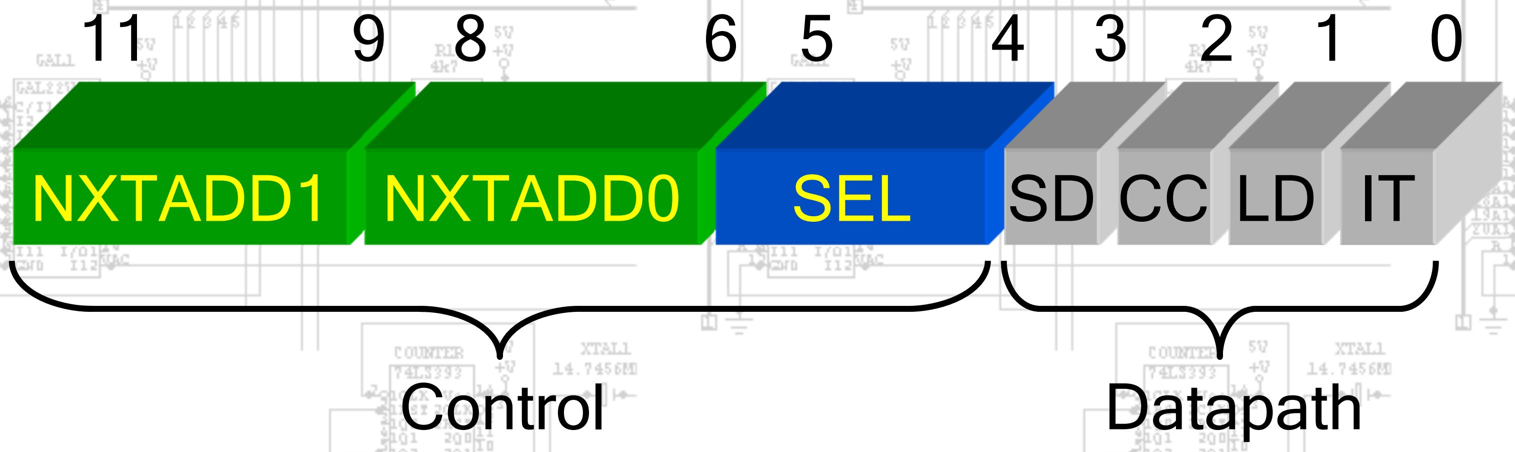
- The control word must supply four control signals:



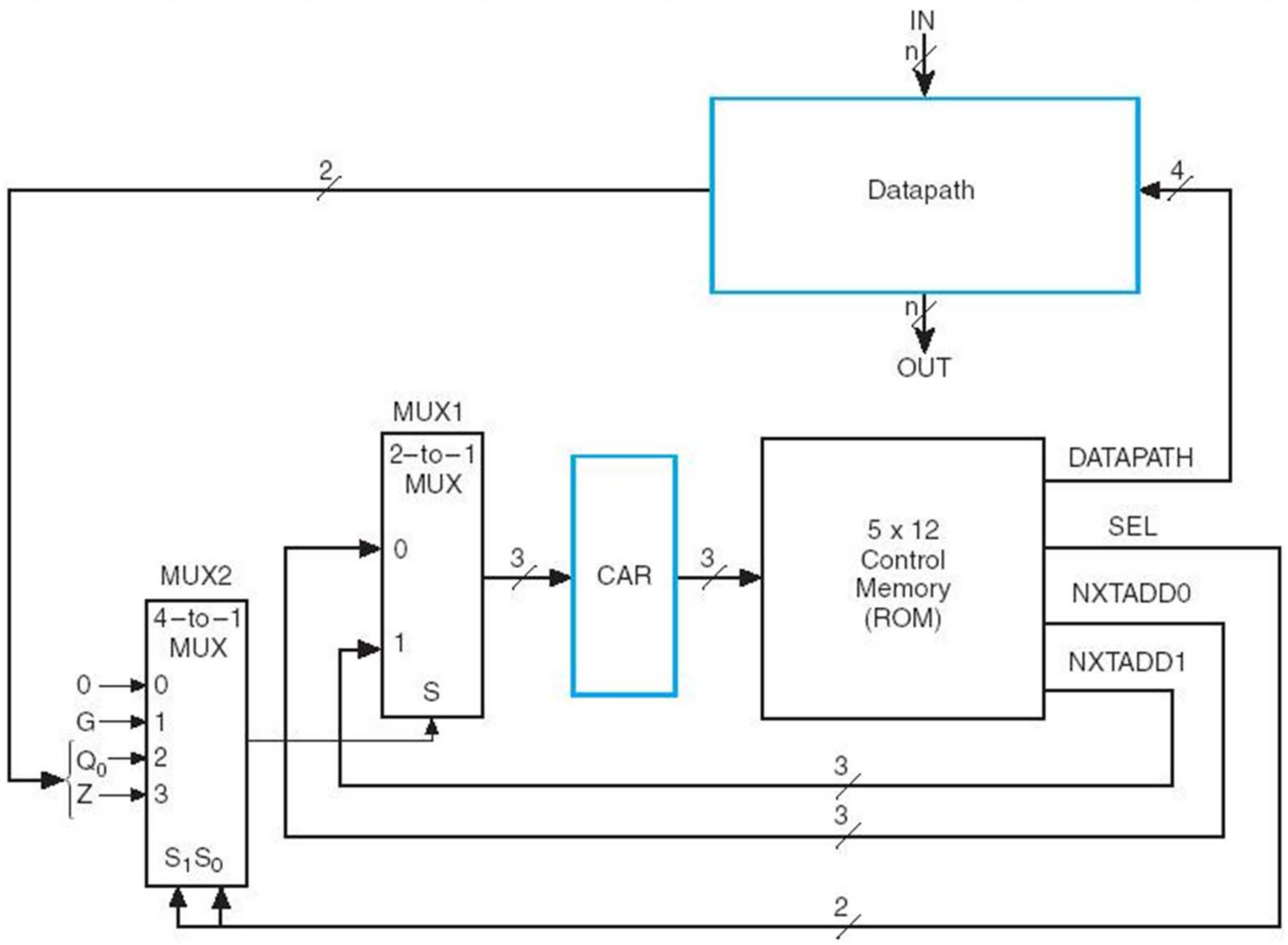
Control Signal	Register Transfers	States in Which Signal is Active	Micro-instruction Bit Position	Symbolic Notation
Initialize	$A \leftarrow 0, P \leftarrow n-1$	INIT	0	IT
Load	$A \leftarrow A+B, C \leftarrow C_{out}$	ADD	1	LD
Clear_C	$C \leftarrow 0$	INIT, MUL1	2	CC
Shift_dec	$C \Vert A \Vert Q \leftarrow \text{sr } C \Vert A \Vert Q, P \leftarrow P-1$	MUL1	3	SD

CS2022 Control Word

- This results in a 12-bit control word:

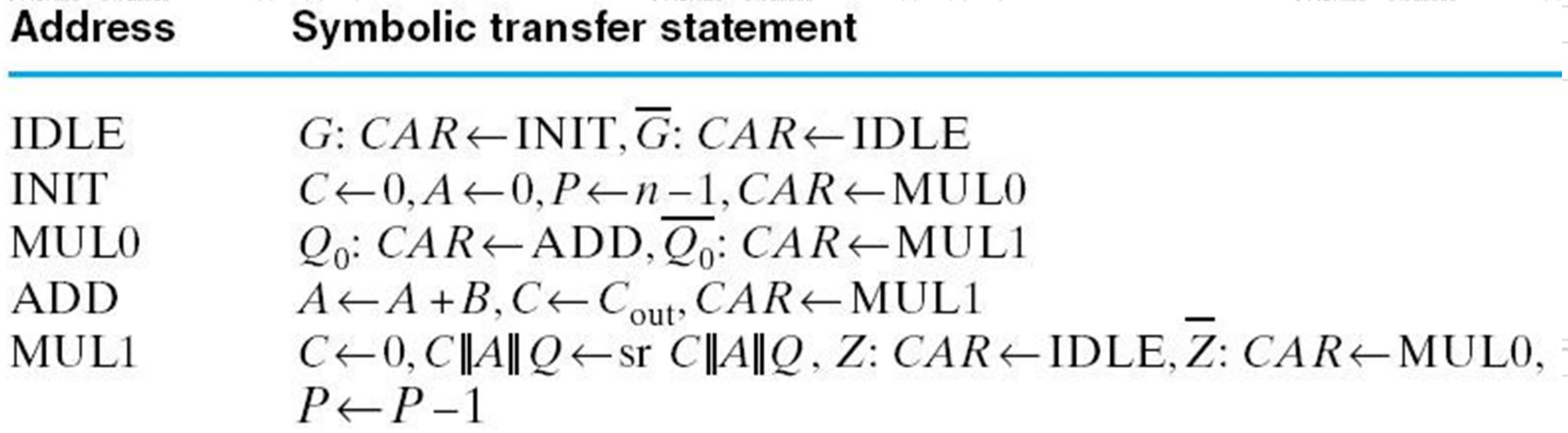


CS2022 Control Unit



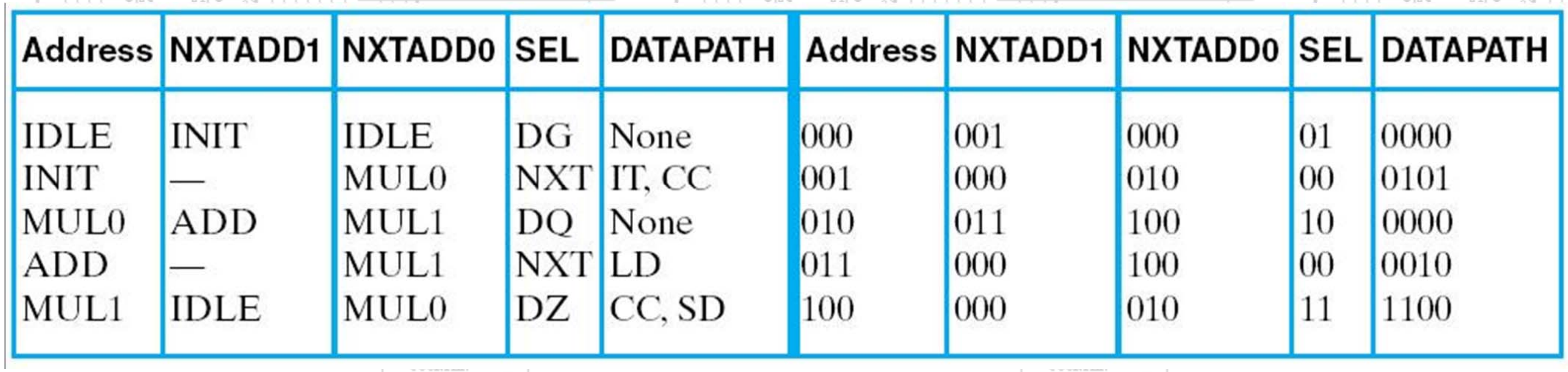
CS2022 Register Transfer Description

- Next we design the microprogram in symbolic RT form:



Address	Symbolic transfer statement
IDLE	$G: CAR \leftarrow \text{INIT}, \overline{G}: CAR \leftarrow \text{IDLE}$
INIT	$C \leftarrow 0, A \leftarrow 0, P \leftarrow n-1, CAR \leftarrow \text{MUL0}$
MUL0	$Q_0: CAR \leftarrow \text{ADD}, \overline{Q_0}: CAR \leftarrow \text{MUL1}$
ADD	$A \leftarrow A + B, C \leftarrow C_{out}, CAR \leftarrow \text{MUL1}$
MUL1	$C \leftarrow 0, C\|A\|Q \leftarrow \text{sr } C\|A\|Q, Z: CAR \leftarrow \text{IDLE}, \overline{Z}: CAR \leftarrow \text{MUL0},$ $P \leftarrow P - 1$

Symbolic Microprogram & Binary Microprogram



Address	NXTADD1	NXTADD0	SEL	DATAPATH	Address	NXTADD1	NXTADD0	SEL	DATAPATH
IDLE	INIT	IDLE	DG	None	000	001	000	01	0000
INIT	—	MUL0	NXT	IT, CC	001	000	010	00	0101
MUL0	ADD	MUL1	DQ	None	010	011	100	10	0000
ADD	—	MUL1	NXT	LD	011	000	100	00	0010
MUL1	IDLE	MUL0	DZ	CC, SD	100	000	010	11	1100