Fundusze Europejskie
Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Społeczny

Politechnika Śląska jako Centrum Nowoczesnego Kształcenia
opartego o badania i innowacje

POWR.03.05.00-IP.08-00-PZ1/17

Projekt współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego

Microprocessor and Embedded Systems

Faculty of Automatic Control, Electronics and Computer Science, Informatics, Bachelor Degree

Lecture 2

8087 numeric co-processor

Bartłomiej Zieliński, PhD, DSc

8087

Program:

- General properties
- A little history – how it all began
- 8086/8087 central unit
- 8086/8087 cooperation rules
- 8087 internal structure
- Registers, data types
- Error control
- Command list

8087

- Basic properties
  - 20 address bits, 8/16 data bits
  - Multiplexed address/data bus
  - Can cooperate with 8086 or 8088, 80186, 80188
    - Automatic detection
  - Abt. 100 times faster floating point operations
  - Operations:
    - Arithmetical (+, -, *, :, √)
    - Comparison
    - Trigonometrical
    - Exponential
    - logarythmical

8087

- General remarks
  - Coprocessor usually shares the system bus with the µp
    - Similarly to DMAC
    - „tightly bounded” multiprocessor system
  - Not fully IEEE 754 standard compatible
    - Standard appeared later than 8087

8087

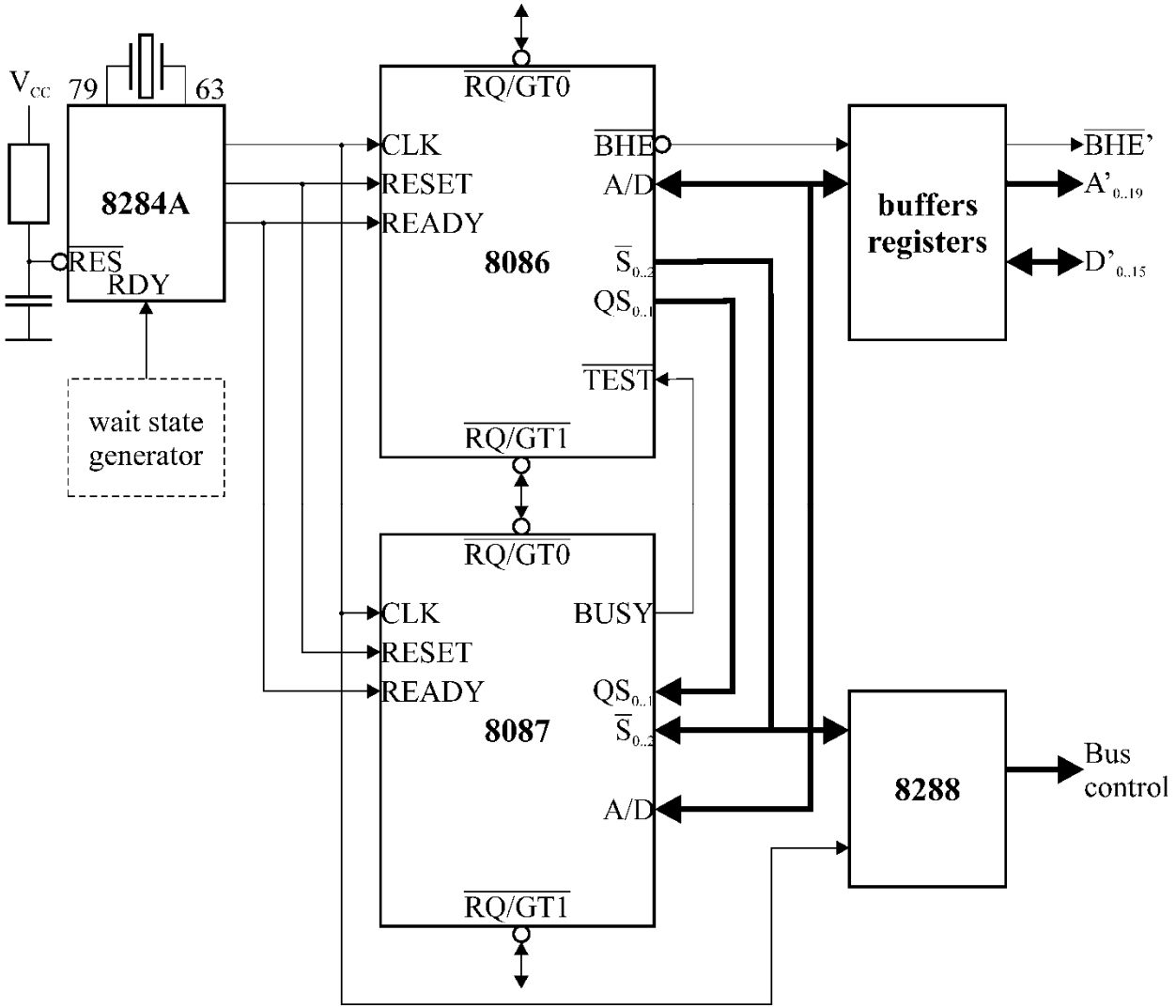
- A bit of history
  - 8231 and 8232 (licensed Am9511, Am9512) (1977-1979)
  - APU (Arithmetic Processing Unit) as I/O device
  - 8-b data bus
  - Cooperation with µp using interrupts or DMA
  - Could work with possibly any µp, e.g.:
    - 8080, 8085, Z-80, Z-8000
    - 6800, 6502 etc.
  - Command execution in paralel to µp possible

8087

- A bit of history (2)
  - Data types
    - 16-b and 32-b fixed point
    - 32-b floating point (later also 64-b floating point)
  - Data stack
    - 8×16-b values
    - 4×32-b values
  - Operated as a typical I/O device
    - Data R/W, command write, status read
  - Long execution times ☹

8087

- 8086/8087 central unit



8087

- 8086/8087 cooperation rules

  - Idea:
    - A programmer thinks it's a single IC, but:
      - More commands (extended command set)
      - More registers (including coproc. registers)
      - New data types (including coproc. data types)
    - Ease of programming
      - Common, uniform, readable code

8087

- 8086/8087 cooperation rules
  - Implementation
    - 8087 determines 8086 state using S0..2 and QS0..1
    - Traces and decodes commands synchronously with 8086 (no action from 8086 required)
    - Fetches every command, executes only its own cmd's
    - If 8087 command is found, starts independent work
      - 8086 can continue if no more 8087 command are present
      - Otherwise it waits until 8087 command is executed
    - 8087 error → interrupt (usually using 8259A)
    - Data transfer – bus acquisition using RQ/GT mechanism

8087

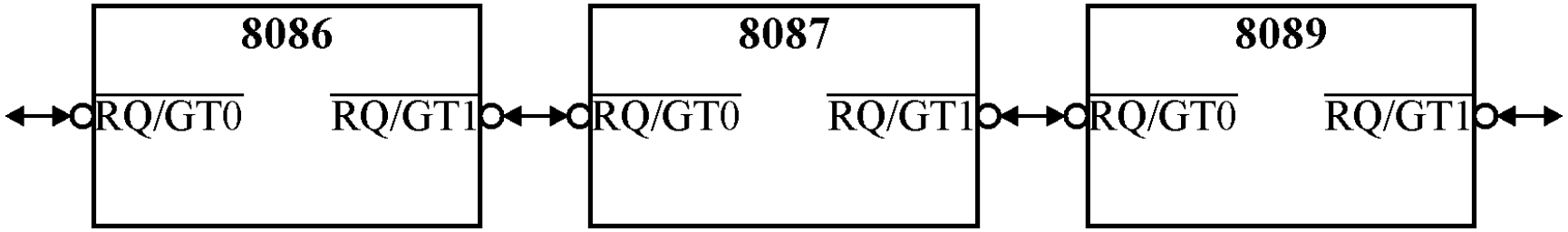
- 8086/8087 cooperation rules
  - If FPU command has memory arguments
    - Read
      - 8086 calculates address
      - „dummy read”
        - » Data ignored by 8086, read by 8087
        - » Address stored in 8087
      - argument size > 16b → consecutive cycles (RQ/GT)
    - Write
      - 8086 calculates address
      - „dummy read”
        - » Data ignored by 8086 and 8087
        - » Address stored in 8087
      - Data written during consecutive cycles (RQ/GT)

8087

- 8086/8087 cooperation rules
  - FPU must know bus size (8/16 bits)
    - How to recognize 8086 and 8088 µp's?
    - Pin 34 right after power-on
      - 8086: $\overline{\mathrm{BHE}}$=0
      - 8088: $\overline{\mathrm{SSO}}$=1
    - 8086: 16-b data access if possible
    - 8088: always 8-b data access
    - Bus size has no effect on calculations

8087

- 8086/8087 cooperation rules
  - More coprocessors possible
    - e.g. 8089 – I/O coprocessor
    - Bus control given with 1 clk delay



8087

- 8086/8087 cooperation rules
  - Synchronisation
    - WAIT command between 2 FPU commands
      - Immediately after FPU command
        - » No parallel work
        - » No data conflict
      - Immediately before next FPU command
        - » Parallel work possible
        - » Programmer must ensure no data conflict occurs
          - 8086 does not alter 8087 data in memory
      - In 80286+ µp's, WAIT is automatically executed
        - » Command is decoded but not necessary

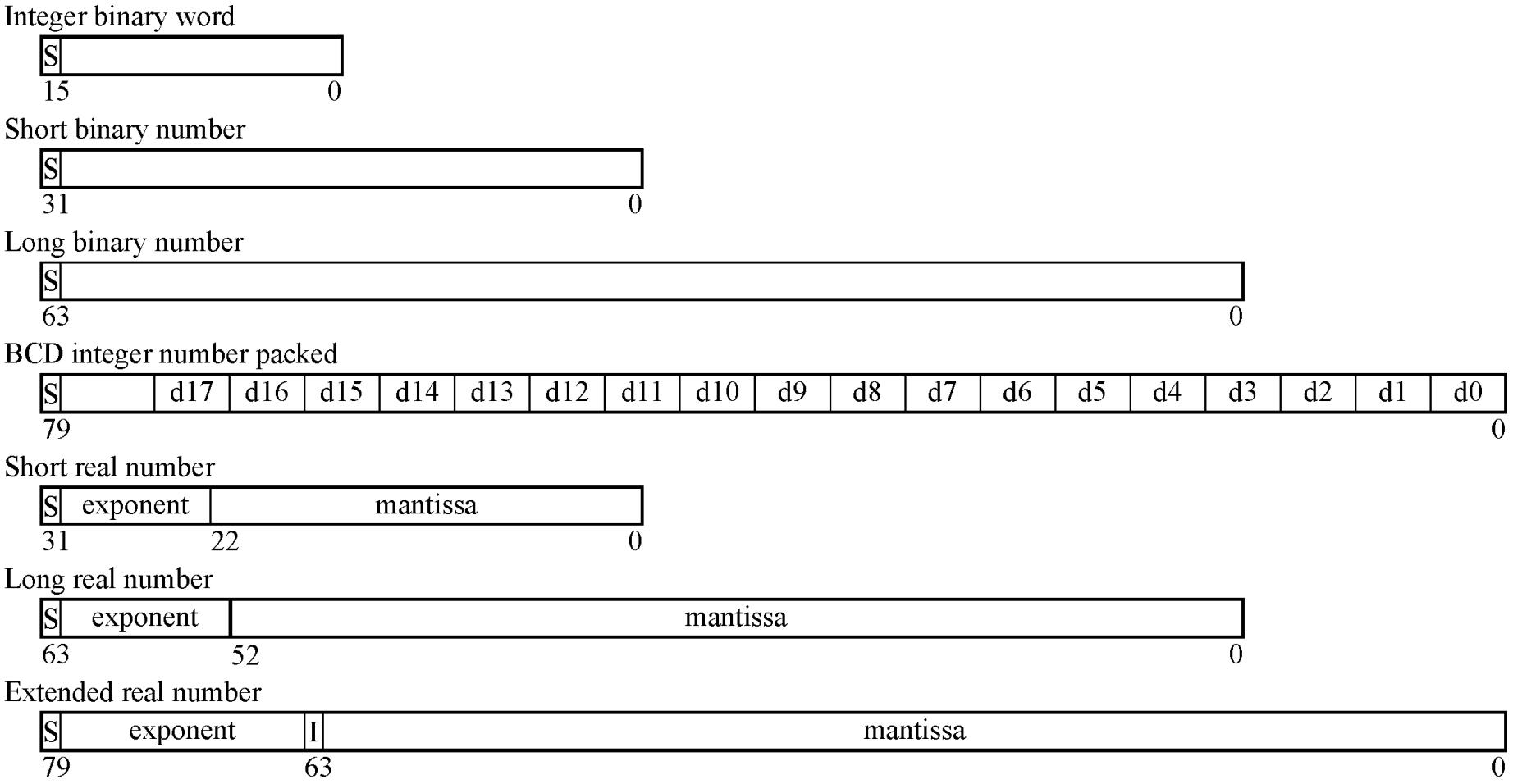
```
FPU command
wait
FPU command

FPU command
wait
CPU commands
FPU command

FPU command
CPU commands
wait
FPU command
```

8087

- 8087 data types
  - Data formats



8087

- 8087 data types
  - Data ranges

Data type	Bits	Significant digits	Min	Max
Integer word	16	4	-32768	32767
Short integer	32	9	-2×10^{9}	2×10^{9}
Long integer	64	18	-9×10^{18}	9×10^{18}
Decimal	80	18	-99…99	99…99
Short real	32	6-7	$\pm1.2\times10^{-38}$	$\pm3.37\times10^{38}$
Long real	64	15-16	$\pm2.3\times10^{-308}$	$\pm1.67\times10^{308}$
Extended real	80	19	$\pm3.4\times10^{-4392}$	$\pm1.2\times10^{4392}$

8087

- 8087 data types
  - Integer types
    - Radix-complement notation
    - 16-b integer as in 8086
  - BCD type
    - Packed – 2 decimal digits in a byte
    - Sign-magnitude notation

8087

- 8087 data types
  - Real types
    - $X = (-1)^S \times M \times 2^E$
    - Normalised numbers →integer of M is 0
      - Higher accuracy
    - Short and long only in memory
      - Automatically converted to extended when loaded to 8087

8087

- 8087 data types
  - Special values
    - Denormal values
      - Exponent too low to be represented in a given format
        - » E.g., exp=-130 too low for short real format
      - Underflow error is masked
      - Calculations can be continued, with possible lower accuracy
      - When calculation error can't be neglected, calculation error is signaled
      - Denormalisation may lose significant 1's at rightmost positions
        - » Number → 0
      - Complemented exponent is 0
      - Integer part of mantissa is 0
        - » Visible only in extended format

8087

- 8087 data types
  - Special values
    - Pseudonormal values
      - Only in extended format
      - Immediate result of masked reaction to underflow
        - » Denormalised → pseudonormal when stored in regs.
      - Exponent is normalised, >0
      - Integer of mantissa is 0
      - Marked as correct in the tag word

      - Don’t stop calculations
      - May be continued despite underflow

8087

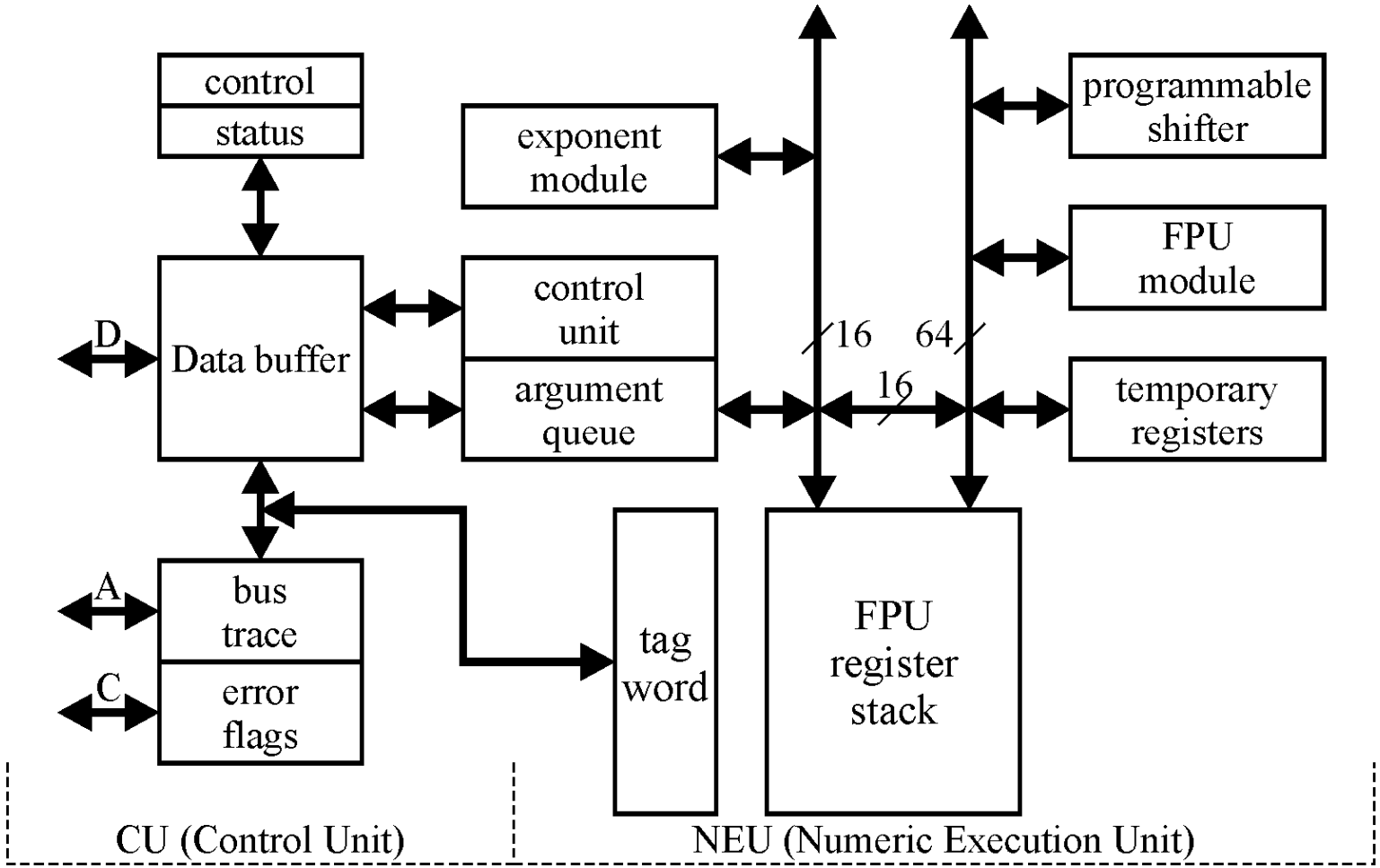
- 8087 data types
  - Special values
    - Zero's and pseudozero's
      - In some types, zero can be positive or negative
        - » But it's still zero and behaves like zero
        - » FXAM can check zero sign
      - Extended type → pseudozero
        - » Pseudonormal, mantissa=0, exponent≠0

8087

- 8087 data types
  - Special values
    - Infinity
      - Marked as such in tag word
      - Result of masked reaction to overflow or division by 0
    - NAN (not a numer)
      - In real types only
      - Marked as such in tag word
      - Result of masked reaction to illegal operation

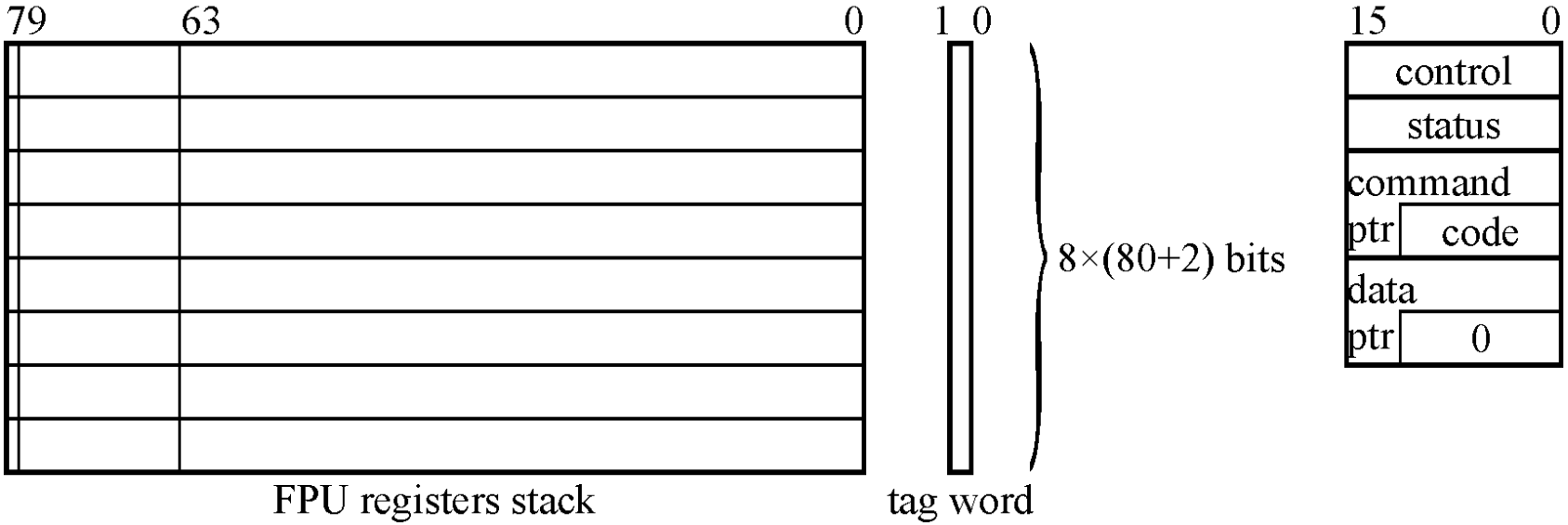
8087

- 8087 structure



8087

- 8087 registers



  - Tag
    - Valid (normalised or pseudonormal)
    - Zero (true zero)
    - Special (NAN, infinity, denormal)
    - empty

8087

- 8087 registers
  - Control
    - Error masks:
      - IM – invalid operation
      - DM – Denormal argument
      - ZM – division by zero
      - OM – overflow
      - UM – underflow
      - PM – precision
    - M – interrupt mask (1 = masked)
    - PC – precision control (24/53/64 mantissa bits)
    - RC – round control (to nearest, down, up, cut)
    - IC – infinity control (unsigned/signed)

8087

- 8087 registers
  - Status:
    - Error flags:
      - IE – invalid operation
      - DE – Denormal argument
      - ZE – division by zero
      - OE – overflow
      - UE – underflow
      - PE – precision
    - IR – interrupt request
    - TOP – top of stack
    - B – busy
    - $C_{0..3}$ – conditional codes

8087

- 8087 registers
  - Status conditional codes
    - Comparison (FCOM), test (FTST)

$C_{3..0}$	FCOM	FTST
00-0	ST > source arg.	ST > 0
00-1	ST < source arg.	ST < 0
10-0	ST = source arg.	ST = ±0
11-1	ST=NAN or ∞ (can't compare)	

8087

- 8087 registers
  - Status conditional codes
    - Examination (FXAM)
      - ST>0, normal
      - ST<0, normal
      - ST>0, pseudonormal
      - ST<0, pseudonormal
      - ST>0, denormalised
      - ST<0, denormalised
      - ST=+NAN
      - ST=-NAN
      - ST=+∞
      - ST=-∞
      - ST=+0
      - ST=-0
      - ST is empty

8087

- 8087 registers
  - Status conditional codes – conditional jumps
    - Example: compare floating point A and B

```
FILD A          ; load A to top of the stack
FCOMP B         ; compare with B
FSTSW st        ; store state
FWAIT           ; wait until ready
MOV AH, st+1    ; copy state MSB to AH
SAHF            ; save state in flag reg.
```

; now condition bits are in 8086 flags

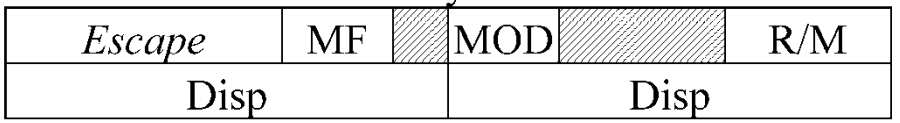
8087

- 8087 error control
  - Invalid operation
    - Write to non-empty register
    - Read from empty register
    - Using argument=NAN
    - Operation on argument that may give unspecified result (e.g., sqrt (negative))
  - Overflow, underflow
    - Result too big/small to fit in the desired format
  - Division by zero
    - Nonzero argument divided by zero
  - Denormalised argument

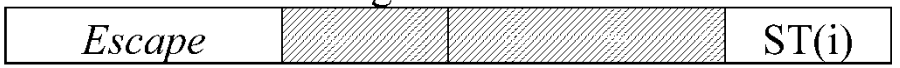
8087

- 8087 commands
  - Example formats
    - MF – number format
      - Real short
      - Integer short
      - Real long
      - Integer word
    - MOD – displacement mode (like in 8086)
    - R/M – addressing mode (similar to 8086)

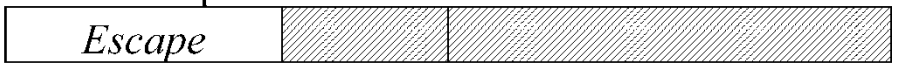
Data transfer to/from memory



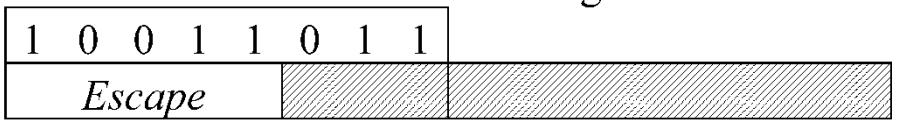
Data transfer between registrers



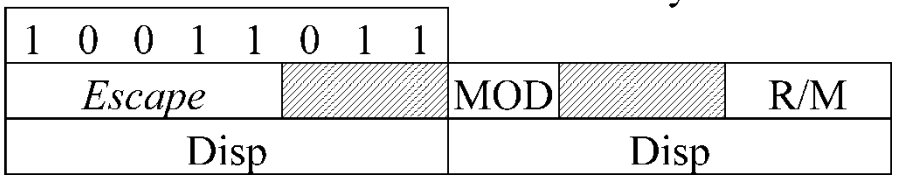
Arithmetical operations



Communication with 8086/88 AX register



Communication with 8086/88 AX memory



8087

- 8087 commands
  - Data transmission
    - FLD – load real number
    - FST/FSTP – store real number (+ pop)
    - FSTP – FST + pop from a stack
    - FILD – load integer number
    - FIST/FISTP – store integer number (+pop)
    - FBLD – load BCD number
    - FBSTP – store BCD number + pop

8087

- 8087 commands
  - Arithmetical (+, -, ×, :)
    - FADD/FADDP – add two real numbers (+ pop)
    - FIADD – add integer numbers
    - FSUB/FSUBP – substract real numbers (+pop)
    - FISUB/FISUBR – substract interger (reverse)
    - FSUBR/FSUBRP – reverse substract real numbers (+ pop)
    - FMUL/FMULP – multiply real numbers (+ pop)
    - FIMUL – multiple integers
    - FDIV/FDIVP – divide real numbers (+ pop)
    - FIDIV/FIDIVR – divide integer numbers (reverse)
    - FDIVR/FDIVRP – reverse divide real numbers (+ pop)

8087

- 8087 commands
  - Arithmetical (cont'd)
    - FSQRT – square root
    - FSCALE – scale by power of 2 (mul/div by power of 2)
    - FPREM – divide remainder
    - FRNDINT – round real to int
    - FXTRACT – extract the exponent and significand
    - FABS – absolute value
    - FCHS – change sign

8087

- 8087 commands
  - Comparison
    - FCOM/FCOMP/FCOMPP – compare (+ pop (+ pop))
    - FICOM/FICOMP – compare integers (+ pop)
    - FTST – compare to zero
    - FXAM – examine stack top

8087

- 8087 commands
  - Functions
    - FPTAN – calculate tangent
    - FPATAN – calculate arctangent
    - F2XM1 – calculate $2^x - 1$
    - FYL2X – calculate $\mathrm{Y} \times \log_2 X$
    - FYL2XP1 – calculate $\mathrm{Y} \times \log_2(X + 1)$

8087

- 8087 commands
  - Constants
    - FLDZ – load zero (+0.0)
    - FLD1 – load 1 (+1.0)
    - FLDPI – load π
    - FLDL2T – load $\log_2 10$
    - FLDL2E – load $\log_2 \mathrm{e}$
    - FLDLG2 – load $\log_{10} 2$
    - FLDLN2 – load $\log_{\mathrm{e}} 2$

8087

- 8087 commands
  - Control (1)
    - FINIT/FNINIT – initialize (software reset)
    - FDISI/FNDISI – disable interrupts
    - FENI/FNENI – enable interrupts
    - FLDCW – load control word from mem
    - FSTCW/FNSTCW – store control word to mem
    - FSTSW/FNSTSW - store status word to mem
    - FCLEX/FNCLEX – clear error flags

8087

- 8087 commands
  - Control (2)
    - FSTENV/FNSTENV – store shortened state to mem
    - FLDENV – restore shortened state from mem
    - FSAVE/FNSAVE – store full state to mem
    - FRSTOR – restore full state from mem
    - FINCSTP – increase stack pointer
    - FDECSTP – decrease stack pointer
    - FFREE – set the tag value to „empty”
    - FNOP – do nothing
    - FWAIT