Interrupts

- Processor Handling
  - software interrupt instruction loaded by processor →
  - processor halts thread execution →
  - processor saves thread state →
    - push return address on the stack
    - push condition register / program status word on the stack
    - put the interrupt routine address in the PC
  - processor executes interrupt handler →
  - processor resumes thread execution
    - pop the condition register / program status word
    - pop the return address to PC

Assembly Language

- Assembly program:
  - use symbolic name of each instruction
  - use label (symbolic address)
  - hardware dependent
  - four elements: label, mnemonic, operand(s), comment
- Sample Assembly Code
# Exercise 1
# This program reads in integers until a negative
# is read. It outputs the sum of the non-negative
# integers.

    MOVEI 0 RB # where the sum will be stored

loop:
    IPORT RA # get the next integer from input to RA
    MOVE RA RA #set N bit of CR if RA is negative
    BRN breakout
    ADD RA RB RB
    BRA loop

breakout:
    OPORT RB # send RB’s value to output port
    EXIT

# Exercise 2
# This program reads in two non-negative integers
# and outputs the product using repeated addition.

    IPORT RA # get first number
    IPORT RB # get the second number
    MOVEI 0 RC # where the product will be stored
    # MOVEI A C: move A to RC where A is an 8-bit 2's complement value

loop:
    ADD RA ZEROS RA #set Z bit of CR if RA == 0, ZEROS 16-bit 0s
    BRZ breakout
    ADD RB RC RC
    ADD RA ONES RA # decrement RA, ONES 16-bit 1s
    BRA loop

breakout:
    OPORT RC
    EXIT