|  |
| --- |
| .data |
|  | counter: .quad 0 |
|  | GPIO: .quad 0 |
|  | .text |
|  | .global setup |
|  | .global interrupt |
|  | #------------------------------------------------- |
|  | # -------------------Interrupt-------------------- |
|  | # ------------------------------------------------ |
|  | setHigh: |
|  | call setGPIO\_high |
|  | decq %rdi |
|  | jmp interruptMask |
|  | interrupt: |
|  | push %rbx |
|  | push %rbp |
|  | incq counter |
|  | movq $4,%rdi # PIN\_NR counter |
|  | movq $1,%rbp # bitmask |
|  | jmp interruptLoop |
|  | interruptMask: |
|  | salq $1,%rbp # Moves the bitmask to the next LED that need to be lit |
|  | interruptLoop: |
|  | cmpq $0,%rdi |
|  | jz interruptEnd # All led statuses has been updated |
|  | movq counter,%rax |
|  | andq %rbp,%rax # Using bitmask operation to find which value the LED is supposed to have. e.g 1101 & 0010 results in 0000 which means that LED 3 should be off |
|  | cmpq $0,%rax # If the submask operation returned a value that isn't zero a bit need to be flipped |
|  | jne setHigh # Sets the required pin to high |
|  | call setGPIO\_low# If the bit is zero then the LED should be off |
|  | decq %rdi # One LED less that hasn't been checked |
|  | jmp interruptMask |
|  | interruptEnd: |
|  | pop %rbp |
|  | pop %rbx |
|  | ret |
|  | #------------------------------------------------- |
|  | # ---------------------Setup---------------------- |
|  | # ------------------------------------------------ |
|  | setup: |
|  | call setGPIO\_adrr # Get base adress |
|  | xor %rsi,%rsi |
|  | xor %rdi,%rdi |
|  | movq $1,%rsi # PIN\_NR counter |
|  | call setGPIO\_pin |
|  | movq $0,%rsi |
|  | movq $1,%rdi |
|  | setupLoop: |
|  | push %rdi |
|  | call setGPIO\_pin # Set GPIO\_PIN nr X (%rdi) to input or output (determined by %rsi) |
|  | pop %rdi |
|  | incq %rdi |
|  | cmpq $4,%rdi |
|  | jle setupLoop |
|  | setupEnd: |
|  | ret |
|  |  |
|  | #------------------------------------------------- |
|  | # ------------------Subroutines------------------- |
|  | # ------------------------------------------------ |
|  |  |
|  | setGPIO\_adrr: |
|  | movq $0x60,%rdi |
|  | movq $16,%rsi |
|  | movq $0xFFFF,%rdx |
|  | movq $0x1000,%rcx |
|  | movq $0x9E,%r8 |
|  | movq $0xFF,%r9 |
|  | call \_sal # salq %rsi,%rdi |
|  | andq %rcx,%rdx |
|  | notq %r8 |
|  | andq %r9,%r8 |
|  | orq %rdx,%rdi |
|  | orq %r8,%rdi |
|  | movq %rdi, GPIO |
|  | movq %rdi,%rax |
|  | ret |
|  | setGPIO\_pin: # rdi = GPIO PIN NR, rsi = input(1) or output(0), |
|  | movq GPIO,%rax |
|  | cmpq $1,%rsi |
|  | jz setInput |
|  | setOutput: |
|  | # (4-GPIO\_NR)\*2 |
|  | subq $4,%rdi # This will result as the negative of the desired outcome |
|  | negq %rdi # Making things right |
|  | salq $1,%rdi # \*2 |
|  | movq $0x2,%rcx # 0x2 << 6 |
|  | salq $6,%rcx |
|  | push %rdi |
|  | push %rsi |
|  | movq %rdi,%rsi # Moving things around to work with \_sal |
|  | movq %rcx,%rdi |
|  | call \_sal # (0x2 << 6) << ((4-GPIO\_NR)\*2) |
|  | movq %rdi,%rcx |
|  | pop %rsi |
|  | pop %rdi |
|  | orq %rcx,(%rax) # \*(short\*)gpio\_addr |= (0x2 << 6) << ((4-GPIO\_NR)\*2) |
|  | ret |
|  | setInput: |
|  | push %rax # Using divq, so we need to push some stuff on to the stack or it will be overridden |
|  | push %rcx |
|  | xor %rdx, %rdx |
|  | movq %rdi,%rax |
|  | movq $8,%rcx |
|  | div %rcx # Remainder stored in %rdx |
|  | pop %rcx # Bringing back the proper values |
|  | pop %rax |
|  | addq %rdx,%rax # gpio\_addr + (GPIO\_NR/8) |
|  | subq $3,%rdi # (((3-GPIO\_NR)\*2) % 8) |
|  | negq %rdi |
|  | salq $1,%rdi |
|  | push %rax |
|  | xor %rdx,%rdx |
|  | movq %rdi, %rax |
|  | movq $8,%rcx |
|  | divq %rcx |
|  | addq $8,%rdx # (((3-GPIO\_NR)\*2) % 8) + 8 |
|  | movq $0x3,%rdi |
|  | movq %rdx,%rsi |
|  | call \_sal # 0x3 << ((((3-GPIO\_NR)\*2) % 8) + 8) |
|  | pop %rax |
|  | orq %rdi,(%rax) # \*(short\*)gpio\_addr + (GPIO\_NR/8) = 0x3 << ((((3-GPIO\_NR)\*2) % 8) + 8) |
|  | ret |
|  | setGPIO\_high: # rdi = GPIO PIN NR |
|  | push %rdi |
|  | subq $1,%rdi # (8\*(GPIO\_NR-1) |
|  | salq $3,%rdi |
|  | movq %rdi,%rsi |
|  | movq $0x3,%rdi # ((0x3 << 4)+1) |
|  | salq $4,%rdi |
|  | incq %rdi |
|  | call \_sal # ((0x3 << 4)+1) << (8\*(GPIO\_NR-1)) |
|  | push %rax |
|  | movq GPIO,%rax # gpio\_addr + 3 |
|  | addq $3,%rax |
|  | orq %rdi,(%rax) # \*(int\*)gpio\_addr + 3 |= ((0x3 << 4)+1) << (8\*(GPIO\_NR-1)) |
|  | pop %rax |
|  | pop %rdi |
|  | ret |
|  |  |
|  | setGPIO\_low: # rdi = GPIO PIN NR |
|  | push %rdi |
|  | subq $1,%rdi # (8\*(GPIO\_NR-1) |
|  | salq $3,%rdi |
|  | movq %rdi,%rsi |
|  | movq $0xCF,%rdi |
|  | call \_sal # 0xCF << (8\*(GPIO\_NR-1) |
|  | notq %rdi # ~(0xCF << (8\*(GPIO\_NR-1)) |
|  | push %rax |
|  | movq GPIO,%rax # gpio\_addr + 3 |
|  | addq $3,%rax |
|  | andq %rdi,(%rax)# \*(int\*)gpio\_addr + 3 &= ~(0xCF << (8\*(GPIO\_NR-1)) |
|  | pop %rax |
|  | pop %rdi |
|  | ret |
|  | \_sal: # salq %rsi,%rdi, %rdi << %rsi |
|  | push %rsi |
|  | \_salLoop: |
|  | cmpq $0,%rsi |
|  | jz \_salEnd |
|  | salq $1,%rdi |
|  | decq %rsi |
|  | jmp \_salLoop |
|  | \_salEnd: |
|  | pop %rsi |
|  | ret |
|  |  |
|  | \_sar: # salq %rsi,%rdi, %rdi >> %rsi |
|  | push %rsi |
|  | \_sarLoop: |
|  | cmpq $0,%rsi |
|  | jz \_salEnd |
|  | sarq $1,%rdi |
|  | decq %rsi |
|  | jmp \_salLoop |
|  | \_sarEnd: |
|  | pop %rsi |
|  | ret |