ELEC6234
Embedded Processor Synthesis

PicoMIPS architecture

- ALU instructions operate on registers only, not RAM
  - ADD, SUB, AND, OR, XOR, NOR
  - ADDI, SUBI, ANDI, ORI, XORI, XNOR
- Others
  - MOV, MOVK: move
  - Branches

Program flow control
- J address - unconditional absolute jump
- B EQ, BNE: conditional relative branch: branch if %s-%s = 0
- Other conditional branches
- BNE branch if ![condition]
- Define BLE: // N : BLT %s,%s = BGE %s,%s
- Define BLT // C=1 (Greater, unsigned), BHI %s,%s

RAM related instructions
- SW %s, [address]: load word from RAM into register %s
- LW %s, [address]: load word from register %s into RAM

Others
- 1 format, no operation
- Subroutines: calls, returns: stack instructions - not implemented in our case study

General approach when synthesising on an FPGA – the decoder should only implement the instructions used by the program stored in the program memory
- Hardware efficient approach

picoMIPS – our case study
- Inspired by MIPS, but not a version of MIPS
- Architecture suited for efficient synthesis on FPGAs with synchronous RAM blocks
- User specifies the width of the data bus, typically 8, 16, 32, 64 bits.
- User specifies the instruction set to be implemented in the decoder

PicoMIPS version 1 – no RAM, no branches

Translating C statements to picoMIPS machine code – If statement

- if (a==0) b=1; else b=2;
  \[
  \text{.def } a=20;\quad \text{a is register 20}
  \text{.def } b=21;\quad \text{b is register 21}
  \text{.blt } a, b, \text{Lebe};\quad \text{if a>0 go to Lebe, nb: reg 50 is always 0}
  \text{ldi } \text{#b};\quad \text{load immediate value b to reg 50}
  \text{ldi } \text{#b};\quad \text{load 1 to reg (ADDI to 50), 1}
  \text{j } \text{Land};\quad \text{jump to Land}
  \text{Lebe: } \text{ldi } \text{#b};\quad \text{actual code ADDI to 50, 2}
  \text{Land: } \text{ldi } \text{#b};\quad \text{actual code ADDI to 50, 2}
\]
Translating C statements to picoMIPS machine code - while statement

- $i=0;\ n=10;\ a=10;\ \text{while}\ (i<n)\ {a += a; i++;}$

```
.def \ i = 16
.def \ n = 17
.LDI \ %a, 0
.LDI \ %a, 10
Loop:
   BLO \ %n, %i, Lend;  ; branch to exit loop if Carry set, i.e. n<i
   ADD \ %a, %a
   ADDI \ %i, %1
   J Loop  ; jump to Loop
Lend:
```

Quick exercise:

- Translate the following ‘for’ statement to picoMIPS assembler
  - $\text{for}(i=0; i<n; i++)$

```
.def \ i = 16
.def \ n = 17
...```

Steps in embedded processor synthesis (picoMIPS case study)

- Develop and test ALU with typical operations: ADD, SUB, etc.
- Develop General Purpose Registers (GPR) and test
- Develop a simple Instruction Decoder and test with ALU and GPR
  - In the decoder implement NOP, ADD, SUB instructions, LD
- Develop Program Memory
- Develop Program Counter
- Enhance Instruction Decoder with J and typical conditional branches, e.g., BEQ, BLO, etc.
- Basic processor core is functional at this point
- Possible enhancements:
  - Add data RAM
  - RAM-related instructions eg, LD, ST
  - Develop an I/O port and implement data-in and OUT instructions
  - Add ALU hardware to support MUL, instruction and shift instruction
  - Subroutine calls and return
- Your coursework assignment:
  - Modify the picoMIPS to implement a peer bed program
  - Write a formal report
  - Challenge: the smallest possible implementation on Altera Cyclone IV