INTRODUCTION TO THE TMS320C6000 VLIW DSP

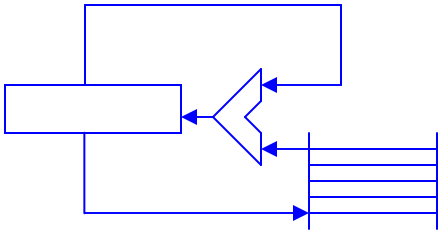
Prof. Brian L. Evans

in collaboration with
Niranjan Damera-Venkata and
Magesh Valliappan

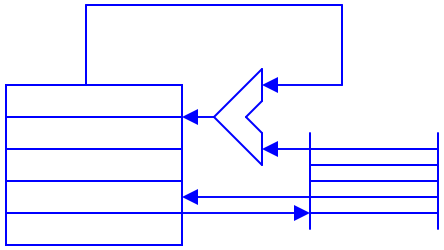
Embedded Signal Processing Laboratory
The University of Texas at Austin
Austin, TX 78712-1084

http://signal.ece.utexas.edu/

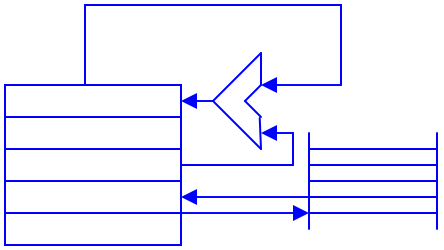
Accumulator architecture



Memory-register architecture



Load-store architecture

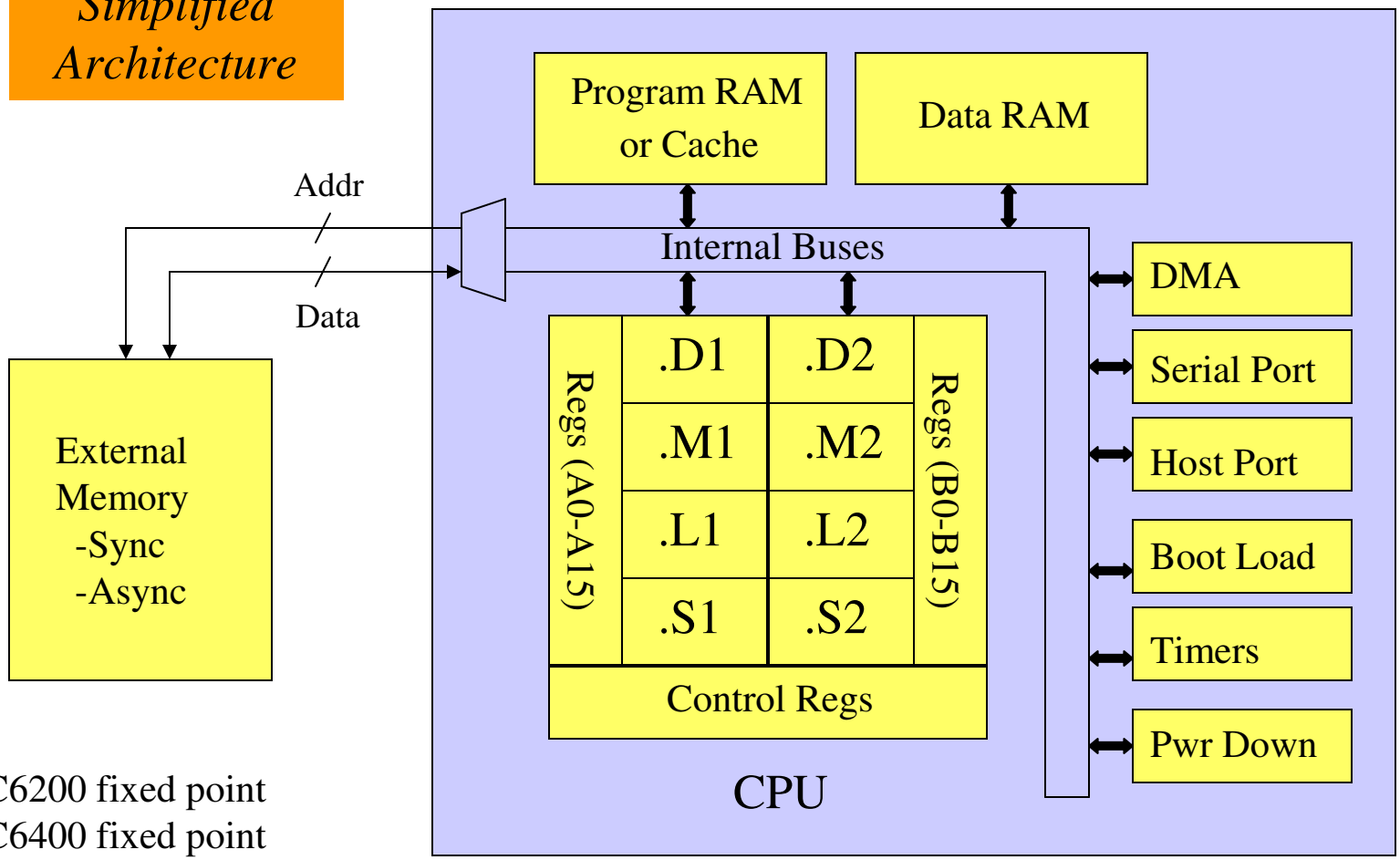


Outline

- C6000 instruction set architecture review
- Vector dot product example
- Pipelining
- Finite impulse response filtering
- Vector dot product example
- Conclusion

TI TMS320C6000 DSP Architecture (Review)

Simplified Architecture



C6200 fixed point
C6400 fixed point
C6700 floating point

TI TMS320C6000 DSP Architecture (Review)

- Address 8/16/32 bit data + 64-bit data on C67x
- Load-store RISC architecture with 2 data paths
  - 16 32-bit registers per data path (A0-A15 and B0-B15)
  - 48 instructions (C6200) and 79 instructions (C6700)
- Two parallel data paths with 32-bit RISC units
  - **D**ata unit - 32-bit address calculations (modulo, linear)
  - **M**ultiplier unit - 16 bit x 16 bit with 32-bit result
  - **L**ogical unit - 40-bit (saturation) arithmetic & compares
  - **S**hifter unit - 32-bit integer ALU and 40-bit shifter
  - Conditionally executed based on registers A1-2 & B0-2
  - Work with two 16-bit halfwords packed into 32 bits

TI TMS320C6000 DSP Architecture (Review)

- .M multiplication unit
  - 16 bit x 16 bit signed/unsigned packed/unpacked
- .L arithmetic logic unit
  - Comparisons and logic operations (and, or, and xor)
  - Saturation arithmetic and absolute value calculation
- .S shifter unit
  - Bit manipulation (set, get, shift, rotate) and branching
  - Addition and packed addition
- .D data unit
  - Load/store to memory
  - Addition and pointer arithmetic

C6000 Restrictions on Register Accesses

- Each function unit has read/write ports
  - Data path 1 (2) units read/write A (B) registers
  - Data path 2 (1) can read one A (B) register per cycle
- Two simultaneous memory accesses cannot use registers of same register file as address pointers
- Limit of four 32-bit reads per register per cycle
- 40-bit longs stored in adjacent even/odd registers
  - Extended precision accumulation of 32-bit numbers
  - Only one 40-bit result can be written per cycle
  - 40-bit read cannot occur in same cycle as 40-bit write
  - 4:1 performance penalty using 40-bit mode

Other C6000 Disadvantages

- No ALU acceleration for bit stream manipulation
  - 50% computation in MPEG-2 decoder spent on variable length decoding on C6200 in C
  - C6400 direct memory access controllers shred bit streams (for video conferencing & wireless basestations)
- Branch in pipeline disables interrupts:
  Avoid branches by using conditional execution
- No hardware protection against pipeline hazards:
  Programmer and tools must guard against it
- Must emulate many conventional DSP features
  - No hardware looping: use register/conditional branch
  - No bit-reversed addressing: use fast algorithm by Elster
  - No status register: only saturation bit given by .L units

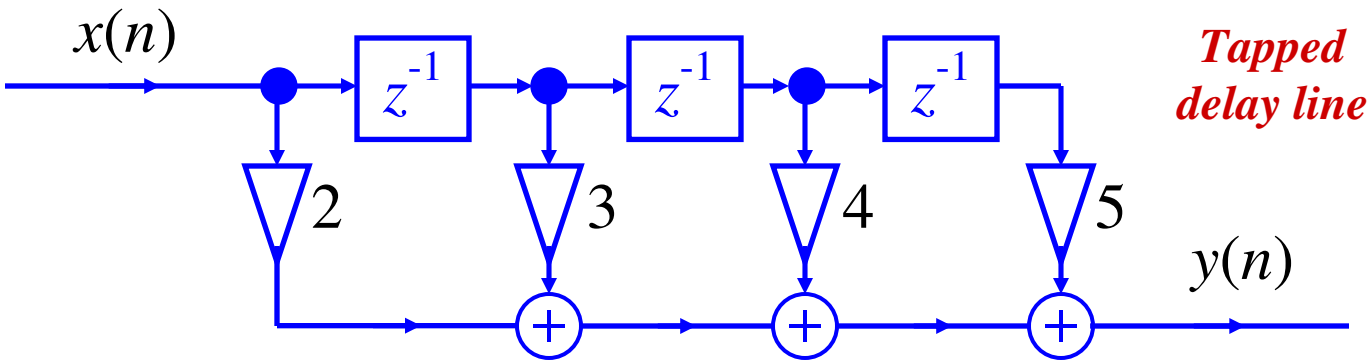
FIR Filter

- Difference equation (vector dot product)

$$y(n) = 2\,x(n) + 3\,x(n-1) + 4\,x(n-2) + 5\,x(n-3)$$

- Signal flow graph

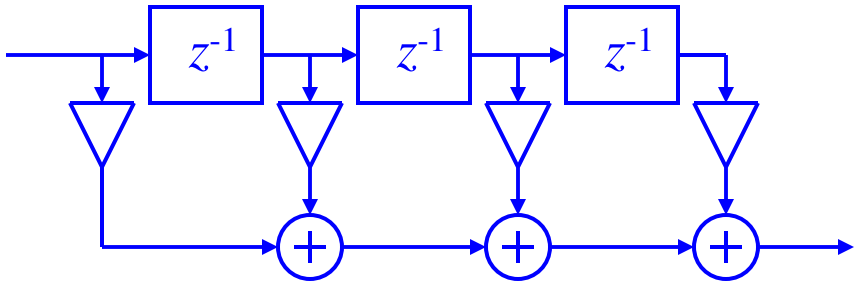
$$y(n) = \sum_{i=0}^{N-1} a(i)\;\; x(n-i)$$



- Dot product of inputs vector and coefficient vector
- Store input in circular buffer, coefficients in array

FIR Filter

- Each tap requires
  - Fetching data sample
  - Fetching coefficient
  - Fetching operand
  - Multiplying two numbers
  - Accumulating multiplication result
  - Shifting one sample in the delay line



- Computing an FIR tap in one instruction cycle
  - Two data memory and one program memory accesses
  - Auto-increment or auto-decrement addressing modes
  - Modulo addressing to implement delay line as circular buffer

Example: Vector Dot Product (Unoptimized)

- A vector dot product is common in filtering

$$Y = \sum_{n=1}^{N} a(n)\ x(n)$$

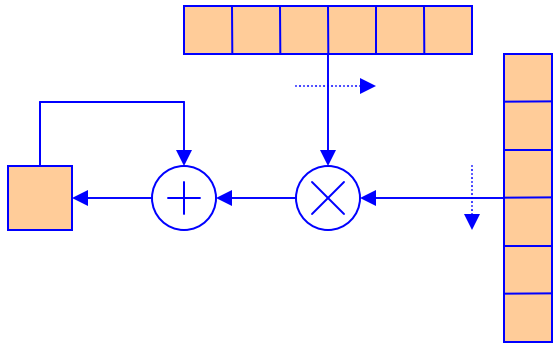
- Store $a(n)$ and $x(n)$ into an array of N elements
- C6000 peaks at 8 RISC instructions/cycle
  - For 300-MHz C6000, RISC instructions per sample: 300,000 for speech; 54,421 for audio CD; and 290 for luminance NTSC video
  - Generally requires hand coding for peak performance
- First dot product example will not be optimized

Example: Vector Dot Product (Unoptimized)

- Prologue
  - Initialize pointers: A5 for $a(n)$, A6 for $x(n)$, and A7 for Y
  - Move the number of times to loop (N) into A2
  - Set accumulator (A4) to zero
- Inner loop
  - Put $a(n)$ into A0 and $x(n)$ into A1
  - Multiply $a(n)$ and $x(n)$
  - Accumulate multiplication result into A4
  - Decrement loop counter (A2)
  - Continue inner loop if counter is not zero
- Epilogue
  - Store the result into Y

Reg	Meaning
A0 A1	$a(n)$ $x(n)$
A2 A3	$N - n$ $a(n)\ x(n)$
A4 A5	Y $\&a$
A6 A7	$\&x$ $\&Y$

Example: Vector Dot Product (Unoptimized)



Using A data path only

A0	$a(n)$
A1	$x(n)$
A2	$N - n$
A3	$a(n)\ x(n)$
A4	Y
A5	& a
A6	& x
A7	& Y

```
; clear A4 and initialize pointers A5, A6, and A7
        MVK   .S1   40,A2       ; A2 = 40 (loop counter)
loop    LDH   .D1   *A5++,A0    ; A0 = a(n)
        LDH   .D1   *A6++,A1    ; A1 = x(n)
        MPY   .M1   A0,A1,A3    ; A3 = a(n) * x(n)
        ADD   .L1   A3,A4,A4    ; Y = Y + A3
        SUB   .L1   A2,1,A2     ; decrement loop counter
[A2]    B     .S1   loop        ; if A2 != 0, then branch
        STH   .D1   A4,*A7      ; *A7 = Y
```

Example: Vector Dot Product (Unoptimized)

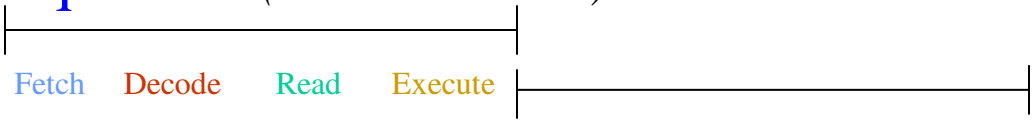
- MoVeKonstant
  - MVK .S 40,A2 ; A2 = 40
  - Lower 16 bits of A2 are loaded
- Conditional branch
  - [condition] B .S loop
  - [A2] means to execute the instruction if A2 != 0
  - Only A1, A2, B0, B1, and B2 can be used
- Loading registers
  - LDH .D *A5, A0 ;Loads half-word into A0 from memory
- Registers may be used as pointers (*A1++)
- Implementation not efficient due to pipeline effects

Pipelining

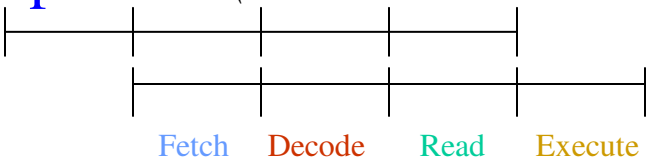
- CPU operations
  - **F**etch instruction from (on-chip) program memory
  - **D**ecode instruction
  - **E**xecute instruction including reading data values
- Overlap operations to increase performance
  - Pipeline CPU operations to increase clock speed over a sequential implementation
  - Separate parallel functional units
  - Peripheral interfaces for I/O do not burden CPU

Pipelining

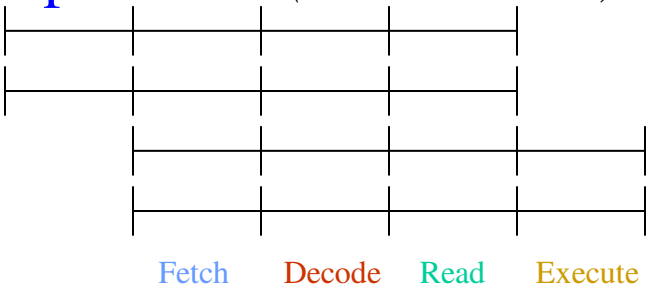
Sequential *(Motorola 56000)*



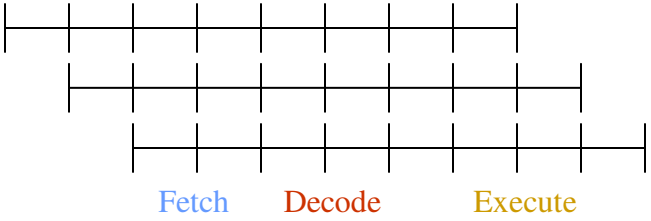
Pipelined *(Most conventional DSP processors)*



Superscalar *(Pentium, MIPS)*



Superpipelined *(TMS320C6000)*



Managing Pipelines

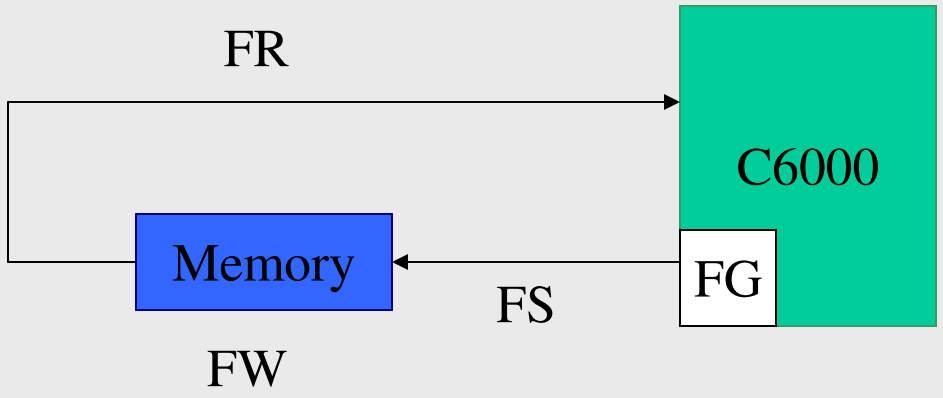
- compiler or programmer (TMS320C6000)
- pipeline interlocking in processor (TMS320C30)
- hardware instruction scheduling

TMS320C6000 Pipeline

- One instruction cycle every clock cycle
- Deep pipeline
  - 7-11 stages in C62x: fetch 4, decode 2, execute 1-5
  - 7-16 stages in C67x: fetch 4, decode 2, execute 1-10
  - If a branch is in the pipeline, interrupts are disabled
  - Avoid branches by using conditional execution
- No hardware protection against pipeline hazards
  - Compiler and assembler must prevent pipeline hazards
- Dispatches instructions in packets

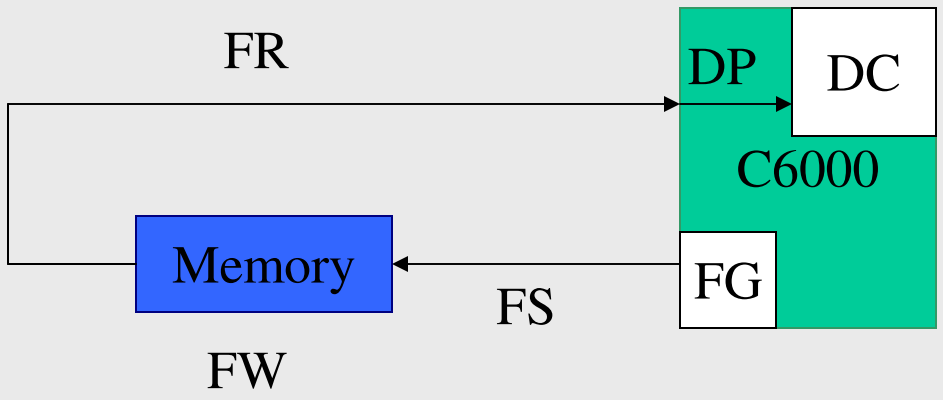
Program Fetch (F)

- Program fetching consists of 4 phases
  - Generate fetch address (FG)
  - Send address to memory (FS)
  - Wait for data ready (FW)
  - Read opcode (FR)
- Fetch packet consists of 8 32-bit instructions



Decode Stage (D)

- Decode stage consists of two phases
  - Dispatch instruction to functional unit (DP)
  - Instruction decoded at functional unit (DC)



Execute Stage (E)

Type	*Description*	*# Instr*	*Delay*
ISC	Single cycle	38	0
IMPY	Multiply	2	1
LDx	Load	3	4
B	Branch	1	5

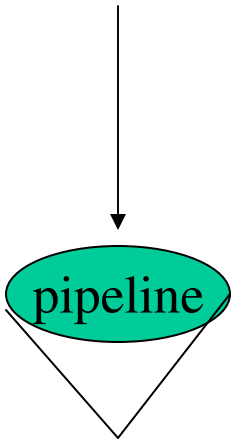
Execute Phase	***Description***
E1	ISC instructions completed
E2	Int. mult. instructions completed
E3	
E4	
E5	Load memory value into register
E6	Branch to destination complete

Vector Dot Product with Pipeline Effects

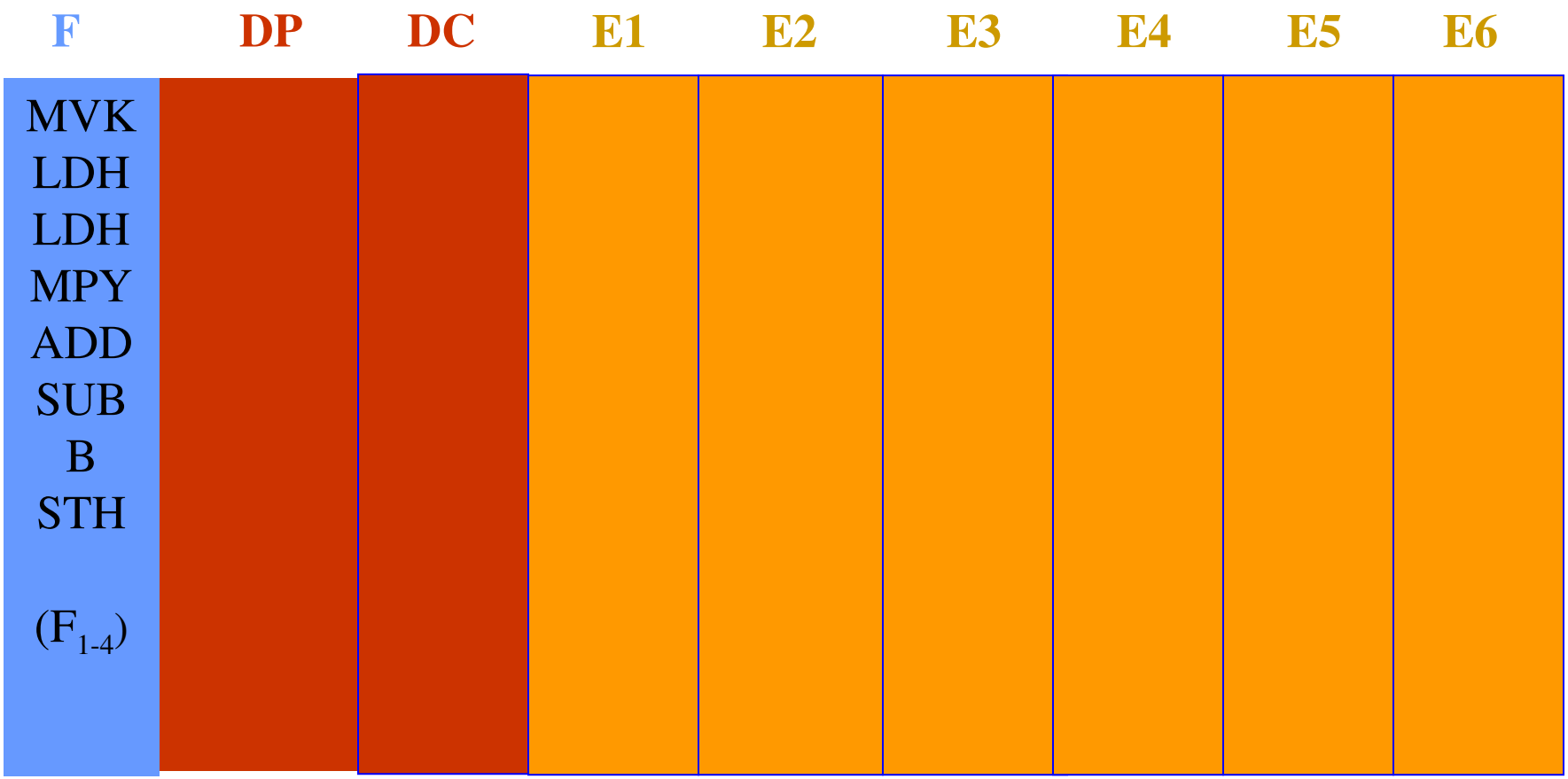
```
; clear A4 and initialize pointers A5, A6, and A7
        MVK   .S1   40,A2       ; A2 = 40 (loop counter)
loop    LDH   .D1   *A5++,A0    ; A0 = a(n)
        LDH   .D1   *A6++,A1    ; A1 = x(n)
        MPY   .M1   A0,A1,A3    ; A3 = a(n) * x(n)
        ADD   .L1   A3,A4,A4    ; Y = Y + A3
        SUB   .L1   A2,1,A2     ; decrement loop counter
[A2]    B     .S1   loop        ; if A2 != 0, then branch
        STH   .D1   A4,*A7      ; *A7 = Y
```

Multiplication has a delay of 1 cycle

Load has a delay of four cycles

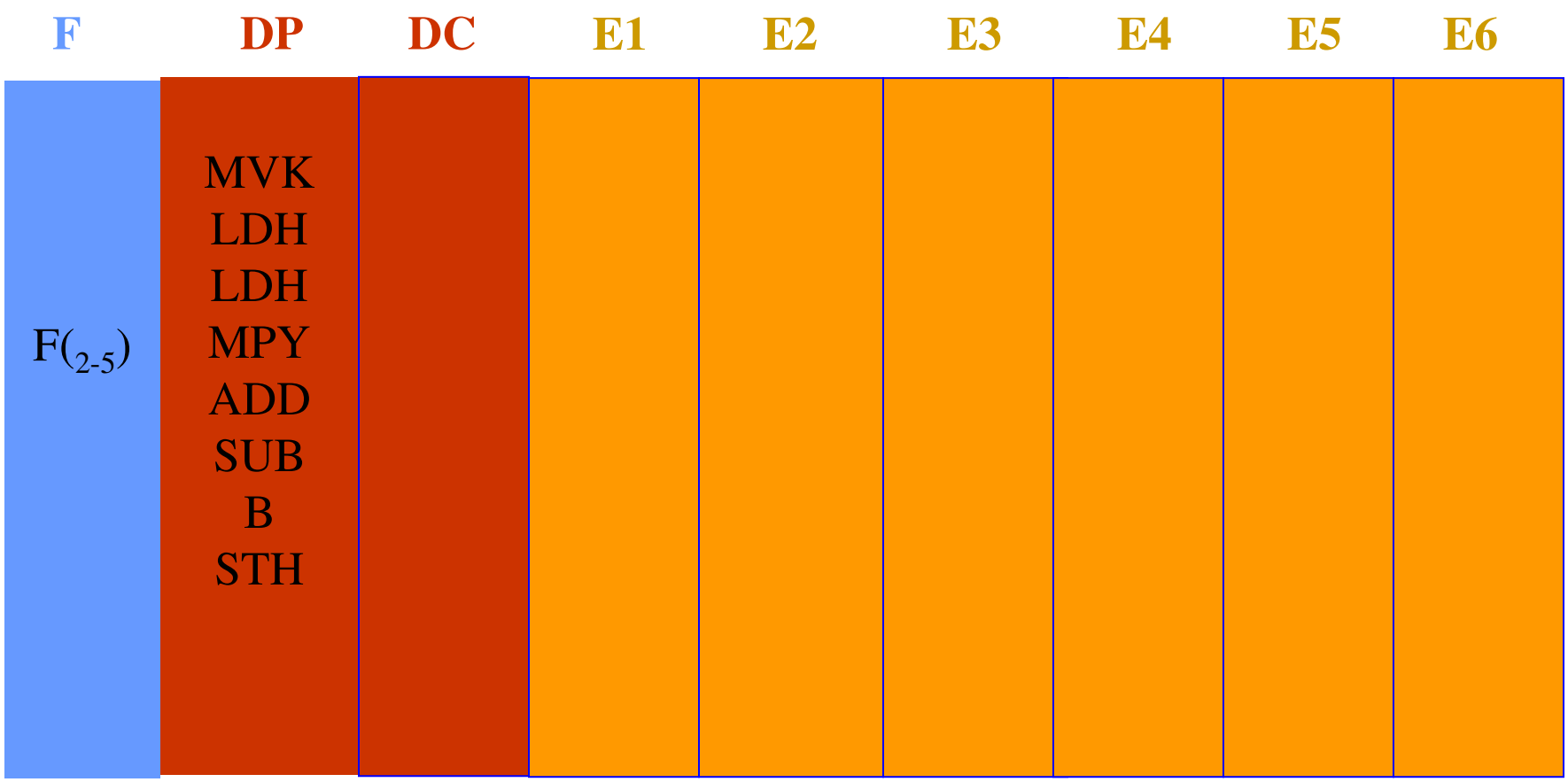


Fetch packet



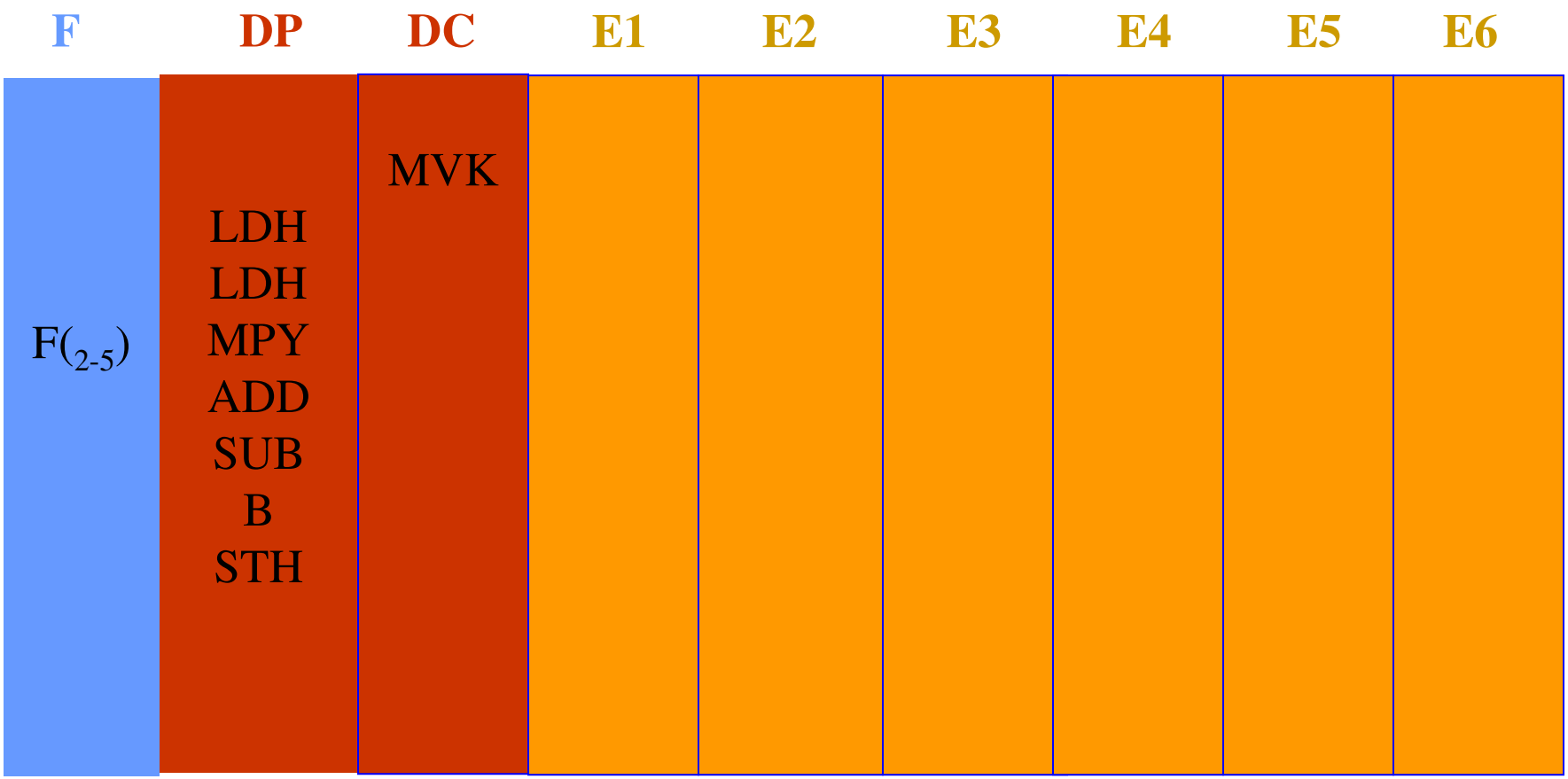
Time (t) = 4 clock cycles

Dispatch



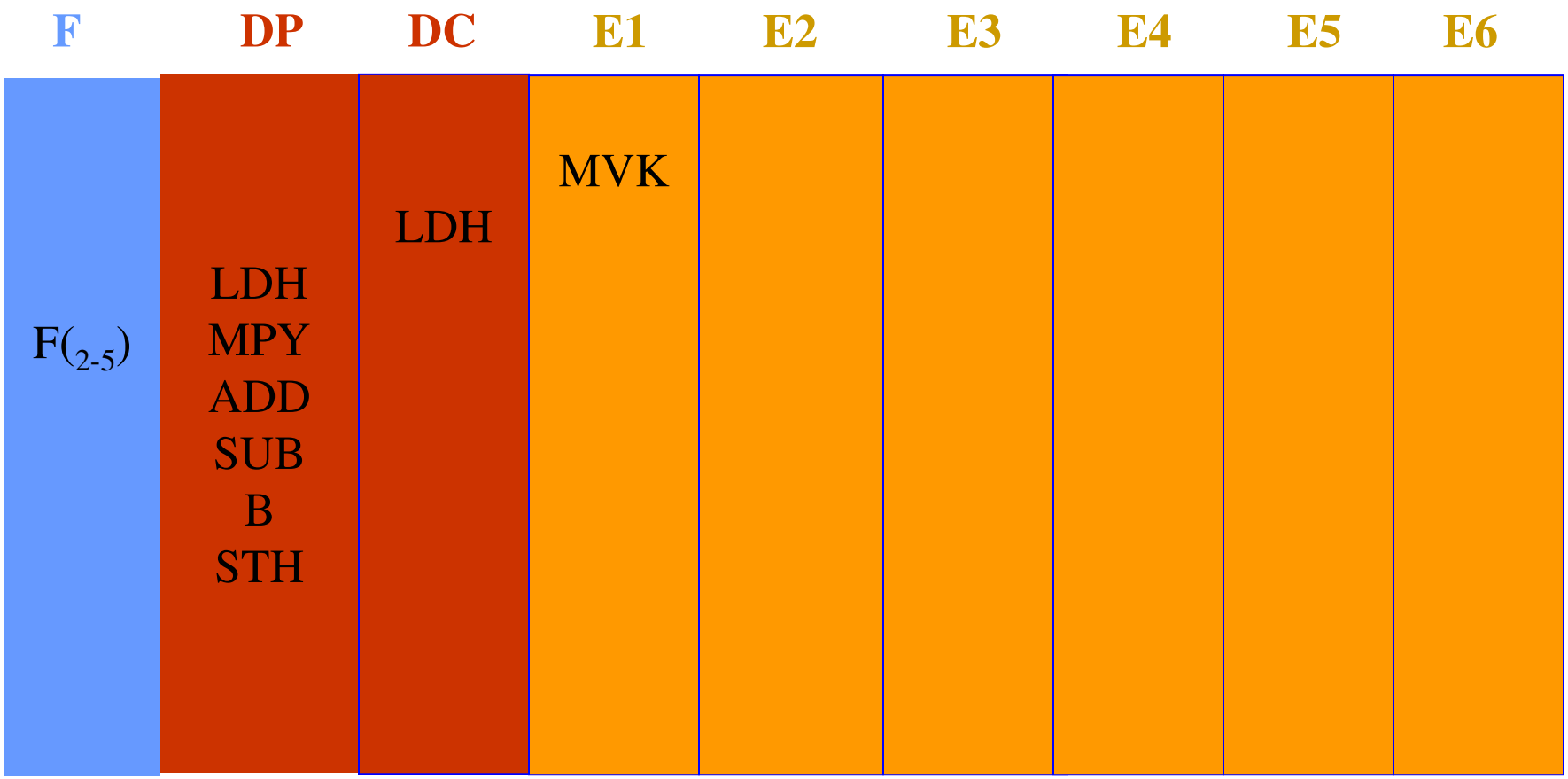
Time (t) = 5 clock cycles

Decode



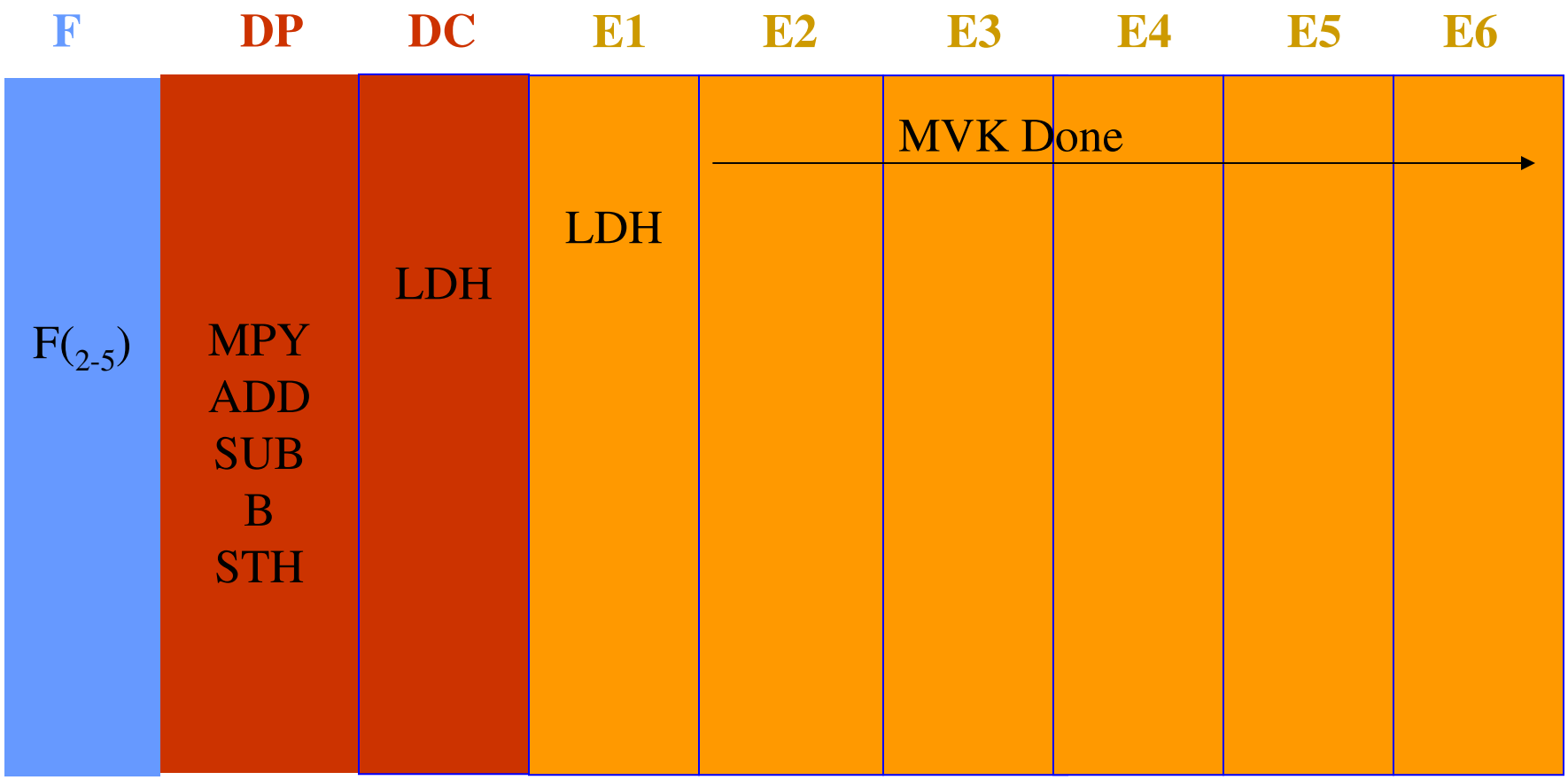
Time (t) = 6 clock cycles

Execute (E1)



Time (t) = 7 clock cycles

Execute (MVK done LDH in E1)



Time (t) = 8 clock cycles

Vector Dot Product with Pipeline Effects

```
; clear A4 and initialize pointers A5, A6, and A7
        MVK   .S1   40,A2       ; A2 = 40 (loop counter)
loop    LDH   .D1   *A5++,A0    ; A0 = a(n)
        LDH   .D1   *A6++,A1    ; A1 = x(n)
        NOP   4
        MPY   .M1   A0,A1,A3    ; A3 = a(n) * x(n)
        NOP
        ADD   .L1   A3,A4,A4    ; Y = Y + A3
        SUB   .L1   A2,1,A2     ; decrement loop counter
[A2]    B     .S1   loop        ; if A2 != 0, then branch
        NOP   5
        STH   .D1   A4,*A7      ; *A7 = Y
```

Assembler will automatically insert NOP instructions

Assembler can also make sequential code parallel

Optimized Vector Dot Product on the C6000

- Prologue
  - Retime dot product to compute two terms per cycle
  - Initialize pointers: A5 for $a(n)$, B6 for $x(n)$, A7 for $y(n)$
  - Move number of times to loop (N) divided by 2 into A2
- Inner loop
  - Put $a(n)$ and $a(n+1)$ in A0 and $x(n)$ and $x(n+1)$ in A1 (***packed data***)
  - Multiply $a(n)$ $x(n)$ and $a(n+1)$ $x(n+1)$
  - Accumulate even (odd) indexed terms in A4 (B4)
  - Decrement loop counter (A2)
- Store result

Reg	Meaning
A0 B1	$a(n) \mid\mid a(n+1)$ $x(n) \mid\mid x(n+1)$
A2	$(N-n)/2$
A3 B3	$a(n)\, x(n)$ $a(n+1)\, x(n+1)$
A4 B4	$y_{even}(n)$ $y_{odd}(n)$
A5 B6 A7	$\&a$ $\&x$ $\&y$

FIR Filter Implementation on the C6000

```
        MVK   .S1 0x0001,AMR ; modulo block size 2^2
        MVKH  .S1 0x4000,AMR ; modulo addr register B6
        MVK   .S2 2,A2       ; A2 = 2 (four-tap filter)
        ZERO  .L1 A4         ; initialize accumulators
        ZERO  .L2 B4
; initialize pointers A5, B6, and A7
fir     LDW   .D1  *A5++,A0  ; load a(n) and a(n+1)
        LDW   .D2  *B6++,B1  ; load x(n) and x(n+1)
        MPY   .M1X A0,B1,A3  ; A3 = a(n) * x(n)
        MPYH  .M2X A0,B1,B3  ; B3 = a(n+1) * x(n+1)
        ADD   .L1  A3,A4,A4  ; yeven(n) += A3
        ADD   .L2  B3,B4,B4  ; yodd(n) += B3
[A2]    SUB   .S1  A2,1,A2   ; decrement loop counter
[A2]    B     .S2  fir       ; if A2 != 0, then branch
        ADD   .L1  A4,B4,A4  ; Y = Yodd + Yeven
        STH   .D1  A4,*A7    ; *A7 = Y
```

Throughput of two multiply-accumulates per instruction cycle

Selected TMS320C6000 Fixed-Point DSPs

DSP	MHz	MIPS	Data (kbits)	Program (kbits)	Level 2 (kbits)	Price	Applications
C6202	250	2000	1000	2000		$ 59	
	300	2400				$ 70	
C6203	250	2000	4000	3000		$ 63	modems banks;
	300	2400				$ 75	ADSL1 modems
C6204	200	1600	512	512		$ 10	
C6416	500	4000	128	128	8000	$ 95	ADSL2 modems
	1000	8000	128	128	8000	$250	3G basestations
C6418	500	4000	128	128	5000	$ 57	
	600	4800	128	128	5000	$ 56	
DM640	400	3200	128	128	1000	$ 23	Video conferencing
DM641	500	4000	128	128	1000	$ 35	Video conferencing
	600	4800	128	128	1000	$ 38	
DM642	500	4000	128	128	2000	$ 43	Video conferencing
	720	5760	128	128	2000	$ 68	

C6416 has Viterbi and Turbo decoder coprocessors.

Unit price is for 1,000 units. Prices effective June 3, 2005.

For more information: http://www.ti.com

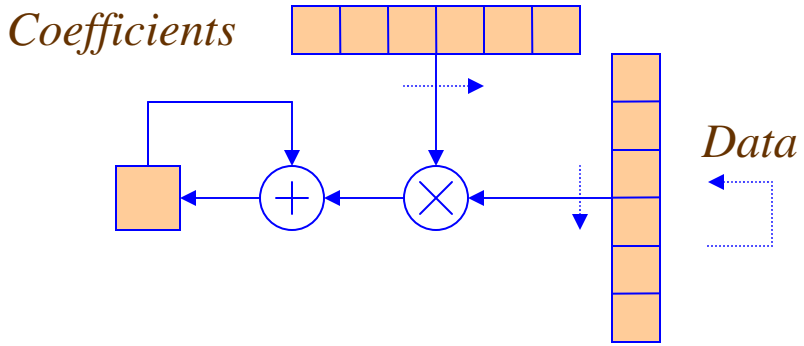
Conclusion

- Conventional digital signal processors
  - High performance vs. power consumption/cost/volume
  - Excel at one-dimensional processing
  - Have instructions tailored to specific applications
- TMS320C6000 VLIW DSP
  - High performance vs. cost/volume
  - Excel at multidimensional signal processing
  - Maximum of 8 RISC instructions per cycle

Conclusion

- Web resources
  - comp.dsp news group: FAQ www.bdti.com/faq/dsp_faq.html
  - embedded processors and systems: www.eg3.com
  - on-line courses and DSP boards: www.techonline.com
- References
  - R. Bhargava, R. Radhakrishnan, B. L. Evans, and L. K. John, "Evaluating MMX Technology Using DSP and Multimedia Applications," *Proc. IEEE Sym. Microarchitecture*, pp. 37-46, 1998.http://www.ece.utexas.edu/~ravib/mmxdsp/
  - B. L. Evans, "EE345S Real-Time DSP Laboratory," UT Austin. http://www.ece.utexas.edu/~bevans/courses/realtime/
  - B. L. Evans, "EE382C Embedded Software Systems," UT Austin.http://www.ece.utexas.edu/~bevans/courses/ee382c/

FIR Filter on a TMS320C5000




```
COEFFP  .set 02000h          ; Program mem address
X       .set 037Fh           ; Newest data sample
LASTAP  .set 037FH           ; Oldest data sample

        …
        LAR AR3, #LASTAP     ; Point to oldest sample
        RPT #127             ; Repeat next inst. 126 times
        MACD COEFFP, *-      ; Compute one tap of FIR
        APAC
        SACH Y,1             ; Store result -- note shift
```

TMS320C6200 vs. StarCore S140

Feature	*C6200*	*S140*
Functional Units	8	16
multipliers	2	4
adders	6	4
other	--	8
Instructions/cycle	8	6 + branch
RISC instructions *	8	11
conditionals	8	2
Instruction width (bits)	256	128
Total instructions	48	180
Number of registers	32	51
Register size (bits)	32	40
Accumulation precision (bits) **	32 or 40	40
Pipeline depth (cycle)	7-11	5

* Does not count equivalent RISC operations for modulo addressing

** On the C6200, there is a performance penalty for 40-bit accumulation